(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号
特開2001－282558
(P2001－282558A)
(43)公開日　平成13年10月12日(2001.10.12)

(51)Int.Cl.[7]	識別記号	ＦＩ		テーマコード(参考)
Ｇ０６Ｆ　9/46	３５０	Ｇ０６Ｆ　9/46	３５０	５Ｂ０９８

審査請求　未請求　請求項の数８　ＯＬ　（全 26 頁）

(21)出願番号　特願2000－95537(P2000－95537)

(22)出願日　平成12年３月30日(2000.3.30)

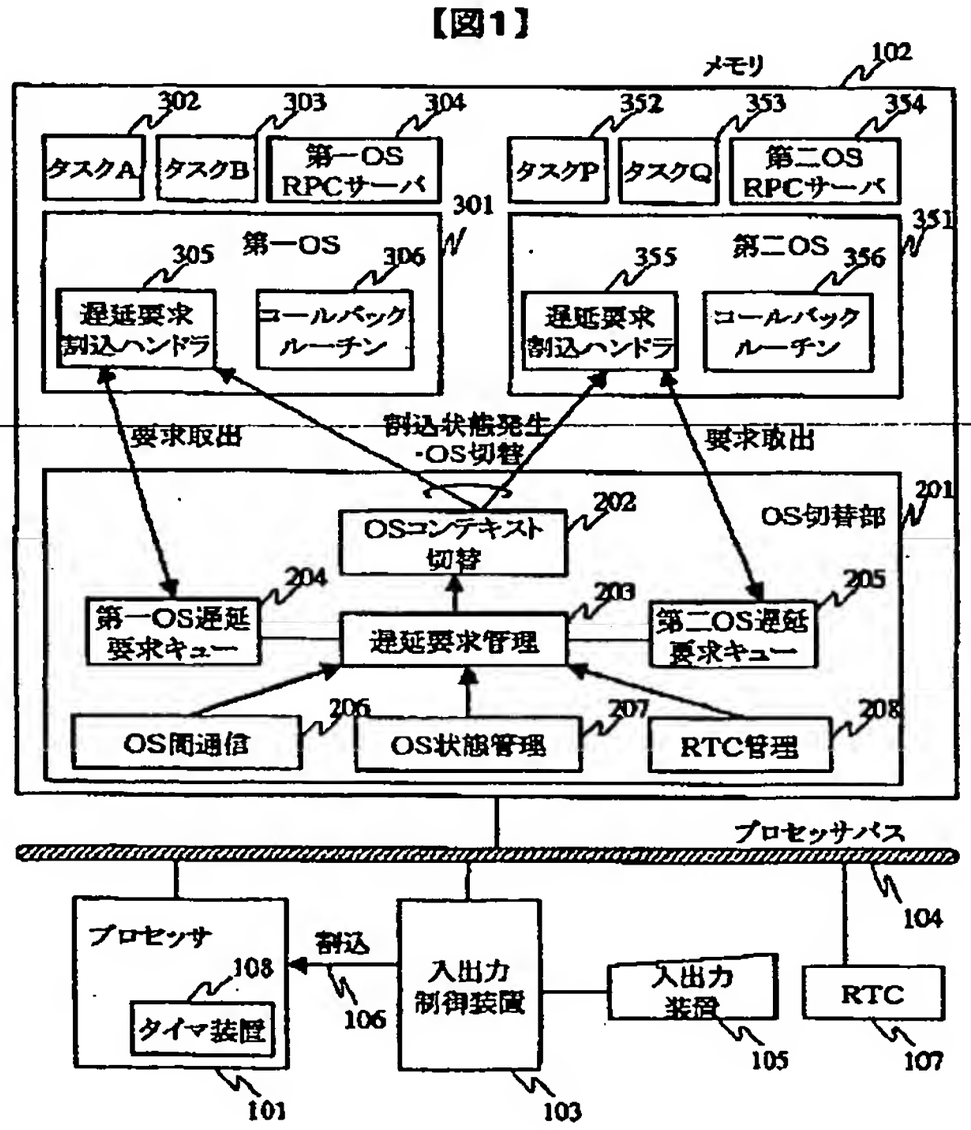
(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地
(72)発明者　大野　洋
茨城県日立市大みか町五丁目２番１号　株式会社日立製作所大みか事業所内
(72)発明者　齊藤　雅彦
茨城県日立市大みか町七丁目１番１号　株式会社日立製作所日立研究所内
(74)代理人　100074631
弁理士　高田　幸彦　（外１名）

最終頁に続く

(54)【発明の名称】　マルチオペレーティング計算機システム

(57)【要約】
【課題】本発明の目的は、複数個のオペレーティングシステム間の処理要求の通信をオペレーティングシステムの簡単な改造で行えるマルチオペレーティング計算機システムを提供することにある。
【解決手段】本発明は、複数個のオペレーティングシステム３０１、３５１間で処理要求を通信によって送信する際に、処理要求を受信したオペレーティングシステムは他のオペレーティングシステムからの処理要求を割込ハンドラ３０５により割込み処理するようにしたことにある。

【図1】



【特許請求の範囲】
【請求項１】複数個のオペレーティングシステムを切替えて一台のプロセッサを動作させ、前記複数個のオペレーティングシステム間で処理要求を行うようにしたマルチオペレーティング計算機システムにおいて、前記処理要求を受けたオペレーティングシステムは他のオペレーティングシステムからの処理要求を割込み処理するようにしたことを特徴とするマルチオペレーティング計算機システム。
【請求項２】複数個のオペレーティングシステムを割込み要求に基づき切替えて一台のプロセッサを動作させ、前記複数個のオペレーティングシステム間で処理要求を行うようにしたマルチオペレーティング計算機システムにおいて、前記処理要求を受けたオペレーティングシステムは他のオペレーティングシステムからの処理要求を割込み処理する割込ハンドラを具備することを特徴とするマルチオペレーティング計算機システム。
【請求項３】複数個のオペレーティングシステムを割込み要求に基づき切替えて一台のプロセッサを動作させ、前記複数個のオペレーティングシステム間で処理要求を行うようにしたマルチオペレーティング計算機システムにおいて、前記オペレーティングシステムが要求する処理要求を一時的に遅延要求キューに格納し、前記処理要求を受けたオペレーティングシステムは他のオペレーティングシステムからの処理要求を前記遅延要求キューから取出す遅延要求割込ハンドラを具備することを特徴とするマルチオペレーティング計算機システム。
【請求項４】割込み要求を発生して複数個のオペレーティングシステムから一つを選択して動作する一台のプロセッサと、前記プロセッサの割込み要求に基づき前記複数個のオペレーティングシステムを切替える切替手段と、前記複数個のオペレーティングシステム間における処理要求を通信する通信手段とを具備し、前記オペレーティングシステムは、他のオペレーティングシステムからの処理要求を割込み処理するようにしたことを特徴とするマルチオペレーティング計算機システム。
【請求項５】割込み要求を発生してオペレーティングシステムを選択して動作する一台のプロセッサと、前記プロセッサの割込み要求に基づき複数個のオペレーティングシステムを選択して切替える切替手段と、前記複数個のオペレーティングシステム間における処理要求を通信する通信手段とを具備し、前記オペレーティングシステムは、他のオペレーティングシステムからの処理要求を割込み処理する割込みハンドラを具備することを特徴とするマルチオペレーティング計算機システム。
【請求項６】割込み要求を発生してオペレーティングシステムを選択して動作する一台のプロセッサと、前記プロセッサの割込み要求に基づき複数個のオペレーティングシステムを選択して切替える切替手段と、前記複数個のオペレーティングシステム間における処理要求を通信する通信手段と、前記オペレーティングシステムが送信した処理要求を一時的に格納する遅延要求キューとを具備し、前記処理要求を受信したオペレーティングシステムは、他のオペレーティングシステムからの処理要求を前記遅延要求キューから取出す遅延要求割込ハンドラを具備することを特徴とするマルチオペレーティング計算機システム。
【請求項７】複数個のオペレーティングシステムを切替えて一台のプロセッサを動作させ、前記複数個のオペレーティングシステム間で処理要求を通信によって送信するようにしたマルチオペレーティング計算機システムにおける前記処理要求を受信したオペレーティングシステムが他のオペレーティングシステムからの処理要求を割込み処理するようにしたプログラムを格納していることを特徴とする記憶媒体。
【請求項８】複数個のオペレーティングシステムを割込み要求に基づき切替えて一台のプロセッサを動作させ、前記複数個のオペレーティングシステム間で処理要求を通信によって送信するようにしたマルチオペレーティング計算機システムにおける前記オペレーティングシステムが送信した処理要求を一時的に遅延要求キューに格納し、前記処理要求を受信したオペレーティングシステムが他のオペレーティングシステムからの処理要求を前記遅延要求キューから取出させるように処理するプログラムを格納していることを特徴とする記憶媒体。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は複数個のオペレーティングシステムを切替えて一台のプロセッサを動作させるマルチオペレーティング計算機システムに関する。
【０００２】
【従来の技術】一つの計算機システムで複数のオペレーティングシステム（以下、「ＯＳ」と呼ぶ）を動作させる技術として、従来から大型計算機において使用されている「仮想計算機（Ｖｉｒｔｕａｌ　Ｍａｃｈｉｎｅ。以下、「VM」と呼ぶ）」が知られている。VMの技術では、計算機システム全体を管理するVMコントロールプログラムが存在し、当該計算機システムのハードウェアを独占して管理する。
【０００３】計算機システム上で動作する各ＯＳに対して、あたかもハードウェアを独占して管理しているのと同様の状態を提供するために、VMコントロールプログラムがハードウェアのエミュレーションを実行する。また、ハードウェア通信路のエミュレーションも提供され、各ＯＳの間に通信路が存在する。
【０００４】複数のＯＳにまたがって通信を行う場合または複数のＯＳにまたがって同期をとる場合（以下、両者を合わせて「ＯＳ間通信」と呼ぶ）には、各ＯＳ上で動作するタスクないし各ＯＳ上で動作する専用のサブプログラムが、通信路を使って通信を行い実現する。各Ｏ

Ｓでこの通信路に対するアクセスが行われるのは、あらかじめ当該ＯＳが実行されるように割当てられたタイムスロットの間だけである。

【０００５】一方、より小規模な計算機システムにおいても、ＣＰＵ能力の向上により、一つの計算機システムで複数のＯＳを動作させることが可能となってきている。このような複数ＯＳ共存技術として特開平１１－１４９３８５が挙げられる。この技術では、ＯＳ切替プログラムが、各ＯＳごとのレジスタ内容等のコンテキストを退避復旧し、ＯＳを切替える。この際、切替の契機となるハードウェア割込についてはＯＳ切替プログラムが取扱うが、それ以外のハードウェアのアクセスについては、動作中のＯＳが実際に操作する。さらに、このような複数ＯＳ共存技術を使った計算機システムにおいて、ＯＳ間通信を実現する方法として、特開平１１－８５５４６にある技術が知られている。

【０００６】

【発明が解決しようとする課題】従来技術のうち、前者のＶＭ技術の場合、全てのハードウェア処理をエミュレーションしており、またＶＭ間の通信路もハードウェアの通信路を模擬したものとなるため、処理のオーバヘッドが大きくなる。従って、ＣＰＵ能力およびメモリ容量等の資源が限られている小規模な計算機システムには適さない。

【０００７】一方、後者の従来技術は、ＯＳ共存技術の場合、当該技術を適用する計算機システムには次のような課題がある。

【０００８】従来技術では、ＯＳを切替えるプログラムが各ＯＳに対して何らかの処理要求を行う場合、当該ＯＳ内の処理モジュールを直接呼出すようにしている。当該計算機システム上で動作するＯＳとして、本来は計算機システム上で単独で動作するＯＳを元にしてこれを修正しようとする場合には、ＯＳ内のタスク切替部分を修正することが必要となる。これは、当該ＯＳのタスク切替に関連する全ての処理を完全に理解することが必要であり、難易度が極めて高いものになる。

【０００９】このことを図３１に示す従来の計算機システムを用いて具体的に説明する。

【００１０】図３１において、第一ＯＳ３０１と第二ＯＳ３５１が、ＯＳ切替部２０１により切替えられながら、単一の計算機システム上で動作している。第一ＯＳ３０１上で動作するタスクＡ３０２は、第二ＯＳ３５１上で動作するタスクＰ３５２からメッセージが送信されるのを待機している。

【００１１】タスクＰからメッセージが送信されたとき、ＯＳ間通信モジュール２０６は第一ＯＳ３０１に対して、待機しているタスクＡ３０２の動作を再開させるよう指示を行う。この際、ＯＳコンテキスト切替モジュール２０２が動作コンテキストを第一ＯＳ３０１のものに復旧した上、コールバックルーチン３０６を呼出す。このときコールバックルーチン３０６は、タスクＡ３０２を再開させる要求をスケジューラ３０９に行う必要がある。

【００１２】しかし、コールバックルーチン３０６は、第一ＯＳ３０１が単独で動作する場合には存在しないものであり、スケジューラ３０９は元々関知していない。従って、コールバックルーチン３０６が呼出される直前に動作していた第一ＯＳ３０１上のタスクのコンテキストを保存する処理や、スケジューラ３０９が当該タスクを中断したものとして扱うような状態変更処理などを、既存のスケジューラの状態管理と矛盾が無いように新たに追加しなければならない。

【００１３】コールバックルーチン３０６が直接スケジューラ３０９を操作せず、タスクＡ３０２を再開させるためのシステムコール３０８を呼出す方法も考えられるが、タスク状態を変更するようなシステムコール３０８の呼出しには様々な制約（例えば、使用するスタックの状態）が伴い、結局ＯＳの内部処理を完全に把握し、その制約を満たす状態を作り出すことが必要になる。

【００１４】また、上述の課題を解決するには図３２に示すように構成することも考えられる。

【００１５】図３２においては、コールバックルーチンは要求されたタスクＡの再開を直接行わず、要求があったことをＯＳ内の要求テーブル３４１に記録し、処理を一旦ＯＳ切替部２０１に戻す。第一ＯＳ上では周期的に起動されるルーチン３４２が存在し、このルーチンが要求テーブル３４１内の要求を取出して、タスクＡ３０２の再開を要求するシステムコール３０８を発行する。この場合、周期起動ルーチン３４２は、ＯＳに組込むデバイスドライバまたはタスクを想定しており、システムコールを発行するための条件が予め整っていてＯＳの内部処理を意識する必要はない。

【００１６】しかし、当該ＯＳに動作が切替っても直前に動作していたタスクが再開するだけであり、周期起動ルーチン３４２が実行されるまでは実際の処理が行われず、タスクＡ３０２の再開が遅延することになる。このため、複数のＯＳを小さいオーバヘッドで切替えて動作させるという本来の目的を達成できなくなる。

【００１７】このように、スケジューラはＯＳ内の全体を管理しているので、他の手順因果関係を考慮した上でＯＳ間通信できるように改造する必要があり、大幅なプログラム変更を要することになる。

【００１８】本発明の目的は、複数個のオペレーティングシステム間の処理要求の通信をオペレーティングシステムの簡単な改造で行えるマルチオペレーティング計算機システムを提供することにある。

【００１９】

【課題を解決するための手段】本発明の特徴とするところは、複数個のオペレーティングシステム間で処理要求を通信によって送信する際に、処理要求を受信したオペ

レーティングシステムは他のオペレーティングシステムからの処理要求を割込み処理するようにしたことにある。

【００２０】本発明は、オペレーティングシステムが処理要求を割込み処理するようにしているので、オペレーティングシステムに通信用のハンドラ機能を一つ追加するだけでよくオペレーティングシステム間の処理要求の通信をオペレーティングシステムの簡単な改造で行える。

【００２１】

【発明の実施の形態】以下、本発明の実施例を図面を用いて説明する。

【００２２】図１に本発明の一実施例を示す。

【００２３】図１において、計算機システムは、プロセッサ１０１、メモリ１０２、入出力制御装置１０３、入出力装置１０５、実時刻時計（Ｒｅａｌ　Ｔｉｍｅ　Ｃｌｏｃｋ。以下、「ＲＴＣ」と呼ぶ）１０７などから構成される。プロセッサ１０１、メモリ１０２、入出力制御装置１０３、ＲＴＣ１０７はプロセッサバス１０４によって接続される。

【００２４】プロセッサ１０１は複数個のオペレーティングシステム（以下、「ＯＳ」と呼ぶ）を動作させるためのマイクロプロセッサである。メモリ１０２は、オペレーティングシステム切替プログラム２０１（以下、「ＯＳ切替部」と呼ぶ）、第一のオペレーティングシステム３０１（以下、「第一ＯＳ」と呼ぶ）、第二のオペレーティングシステム３５１（以下、「第二ＯＳ」と呼ぶ）、第一ＯＳ上で動作するタスク３０２、３０３、第二ＯＳ上で動作するタスク３５２、３５３、第二ＯＳからの要求により第一ＯＳ上での処理を行う第一ＯＳＲＰＣサーバ３０４、第一ＯＳからの要求により第二ＯＳ上での処理を行う第二ＯＳＲＰＣサーバ３５４を保持する。これらのプログラム２０１、３０１～３０４、３５１～３５４は、プロセッサ１０１によって読み出されて実行される。

【００２５】入出力制御装置１０３には、プロセッサ１０１またはメモリ１０２との間で、データを入力または出力するための入出力装置１０５が接続される。入出力装置１０５としては、データを保存するためのディスク装置、画面表示用デバイスであるディスプレイ、ユーザからの入力用デバイスであるキーボード、あるいは他の計算機システムとのデータを交換するためのネットワーク接続装置などがある。どのような入出力装置が接続されるか、あるいは全て接続されないかは、システム構成によって決定される。

【００２６】入出力制御装置１０３は、割込信号線１０６によってプロセッサ１０１と接続され、入出力動作完了などを通知することができる。なお、図１では割込信号線１０６とプロセッサバス１０４が別の装置であるように記載しているが、実際には、割込信号線１０６はプロセッサバス１０４の一部である。プロセッサ１０１の内部にはタイマ装置１０８が設けられており、一定周期毎に割込を発生させる。タイマ装置１０８からの割込は、ＯＳ３０１、３５１の計時などに使用される。

【００２７】なお、プロセッサ１０１の種類によっては、入出力制御装置１０３、入出力装置１０５、ＲＴＣ１０７の一部あるいは全部が、プロセッサ１０１の内部に取り込まれている場合がある。また、タイマ装置１０８がプロセッサ１０１の内部には存在せず、プロセッサバス１０４を介してプロセッサ１０１の外部に接続される場合もある。

【００２８】プロセッサ１０１は割込信号線１０６によって通知されるプロセッサ外部からの割込（以下、「外部割込」と呼ぶ）やタイマ装置１０８などのプロセッサ内部からの割込（以下、「内部割込」と呼ぶ）をマスクできる機能を有している。割込マスクとは、プログラムが割込マスクを解除するまでの間、特定の割込が入ることを遅延させる機能である。

【００２９】一般に、割込マスク機能には、次の三種類が存在する。

（１）全割込マスク：全ての割込をマスクする。

（２）個別割込マスク：個々の割込をそれぞれマスクする。

（３）割込レベルマスク：各割込にレベルを設定し、指定レベル以下の割込をマスクする。

【００３０】プロセッサ１０１の種類によって、上記（１）および（２）の組合せ、または（１）および（３）の組合せのいずれかを装備することが多い。後者の組合せを装備するプロセッサを採用する場合、対応する割込要因の重要性にしたがって割込レベルを割当てる。例えば、リアルタイム性が高いネットワーク入出力装置からの割込を、他のディスク装置などの割込よりも高いレベルに設定する計算機システムの構成がある。

【００３１】本実施例では、計算機システム内に二つのＯＳ３０１、３５１が存在する。ＯＳ３０１、３５１は、各々に割当てられたメモリ・プロセッサ資源を用い、それぞれプログラム３０２～３０４、プログラム３５２～３５４を実行させる。ＲＰＣサーバ３０４ないし３５４は、実際には当該ＯＳ上の二つのタスクから構成される。このように実施例としてはＯＳ数が２、タスク数が８（各ＯＳに対して４ずつ）の例を示しているが、これらの数値よりも多い、または、少ないＯＳないしタスクを実装することも可能である。ＯＳ数は動的にタスク生成・削除を行うことにより可能である。

【００３２】なお、動作している複数のＯＳを全て同等のものとして扱っている。従って、以下で単一のＯＳについてのみしか説明していない部分について、他のＯＳでも同様の構成を持っている、ないし同様の動作が行われるものとする。

【００３３】各ＯＳ３０１、３５１には、ＯＳ切替部２

０１の内部から直接呼出されて要求された処理を完結しＯＳ切替部２０１にリターンする、コールバックルーチン３０６ないし３５６が存在する。ＯＳ切替部２０１は各ＯＳ３０１、３５１に対する処理要求のうち、当該処理要求を即時に実行することが必要なものに関して、コールバックルーチンの呼出を行う。処理要求の例として、当該ＯＳを強制停止させる直前に当該ＯＳが管理する入出力装置からの割込発生を抑止する処理がある。

【００３４】ＯＳ切替部２０１には、ＯＳコンテキスト切替モジュール２０２が設けられており、ＯＳ切替部２０１内の各モジュールからの指示によって、プロセッサ１０１上で動作するＯＳを切替える。また、ＯＳ切替部２０１には、ＯＳ切替部２０１から各ＯＳ３０１、３５１に対する処理要求を管理する遅延要求管理モジュール２０３があり、第一ＯＳ３０１に対する処理要求を蓄積するために第一ＯＳ遅延要求キュー２０４、第二ＯＳ３５１に対する処理要求を蓄積するために第二ＯＳ遅延要求キュー２０５を有している。処理要求の例として、ＯＳ切替部が提供するＯＳ間メッセージ通信機能を使用しメッセージが送られてくるのを受信待機しているタスクがある場合、メッセージが送信された時点で当該タスクの動作を再開させる処理がある。

【００３５】遅延要求管理モジュール２０３は、第一ＯＳ遅延要求キュー２０４ないし第二ＯＳ遅延要求キュー２０５に処理要求が蓄積されていて、なおかつ当該ＯＳが割込を処理するのに適切な条件が整った場合、ＯＳコンテキスト切替モジュール２０２を通じて当該ＯＳに割込が発生したのと同じ状態とし、当該ＯＳに遅延要求の存在を通知する割込を発生させる。以下、遅延要求管理モジュール２０３経由で行われる処理要求を「遅延要求」、遅延要求管理モジュールの存在を通知する割込を「遅延要求割込」と呼ぶ。各ＯＳ３０１、３５１上には、遅延要求割込を受けて、遅延要求キュー２０４ないし２０５から要求を取出し対応する処理を行う、遅延要求割込ハンドラ３０５ないし３５５が設けられている。

【００３６】ＯＳ間通信モジュール２０６は、異なるＯＳのタスク間でのメッセージを交換する機能（以下、「ＯＳ間メッセージ通信」と呼ぶ）や、異なるＯＳ間で同期を取るための機能（以下、「ＯＳ間セマフォ」と呼ぶ）を提供する。ＯＳ間メッセージ通信機能では、メッセージがまだ存在せず送信されるのを待つためにタスクの動作が中断する場合があり、またＯＳ間セマフォ機能では、獲得しようとしたセマフォが先に獲得されていて当該セマフォが解放されるのを待つためにタスクの動作が中断する場合がある。いずれの場合も、待っている条件が整い次第、当該タスクの動作を再開する必要がある。ＯＳ間通信モジュール２０６は前記の条件が整った場合、タスク再開の遅延要求を追加するよう遅延要求管理モジュール２０３に対して依頼する。

【００３７】ＯＳ状態管理モジュール２０７は、各ＯＳ３０１、３５１の起動や停止等の状態を管理し、また状態の移行要求を受付けて各ＯＳ３０１、３５１に指示する機能を有している。例えば、あるタスクからあるＯＳに対するシャットダウン要求を受付けた時は、当該ＯＳに対するシャットダウン処理要求を遅延要求として追加するよう、遅延要求管理モジュール２０３に依頼する。

【００３８】ＲＴＣ管理モジュール２０８は、ＲＴＣ１０７からの時刻取得処理およびＲＴＣ１０７への時刻設定処理を行う。各ＯＳ３０１、３５１は、ＲＴＣ１０７に対して直接時刻を読み書きする代りに、ＲＴＣ管理モジュール２０８にこれらの処理要求を行う。例えば、一方のＯＳ３０１からＲＴＣ１０７への時刻設定処理依頼があった場合、他のＯＳ３５１に対してＲＴＣ１０７に時刻を合わせて修正するよう遅延要求を追加する。

【００３９】各ＯＳ３０１、３５１のＲＰＣサーバ３０４、３５４は、それぞれ、他のＯＳからＯＳメッセージ通信機能を使用して送られてくる処理要求を受付け、当該ＯＳ上で処理を実行し、さらにＯＳメッセージ通信機能を使用して処理結果を送り返す。

【００４０】図２に、プロセッサ１０１の内部構成の一例を示す。

【００４１】図２において、キャッシュメモリ１２２はメモリ１０２上のデータまたは命令を一時的に格納するバッファ記憶装置である。ＣＰＵ１２１は演算手段であり、メモリ１０１もしくはキャッシュメモリ１２２上に存在する命令を順次読み出して実行する。命令実行に際して、演算結果を一時的に保持するための汎用レジスタ１２３、命令アドレスを指定するプログラムカウンタ（以下、「ＰＣ」と呼ぶ）１２４、実行状態を保持するステータスレジスタ（以下、「ＳＲ」と呼ぶ）１２５を用いる。

【００４２】タイマ装置１０８として、三つのタイマ装置１０８１～１０８３を備えている。タイマ装置１０８の個数は、後述の割込処理の説明のために三つとしているが、本発明の実施にあたっては最低一つ存在していればよい。

【００４３】割込信号線１０６とタイマ装置１０８からの割込信号は割込コントローラ１２６に接続される。割込コントローラ１２６は、外部割込または内部割込が発生した場合、ＳＲ１２５の内部にある割込マスクに関する情報を参照し、当該割込を受け付けるか否かを決定する。

【００４４】割込を受付ける場合、割込コントローラ１２６は、割込の要因を割込要因レジスタ１３０に記録し、ＰＣ１２４の現在値を退避プログラムカウンタ（以下、「ＳＰＣ」と呼ぶ）１２６に保存してから割込アドレスレジスタ１２９の値に書換える。さらに、ＳＲ１２５の現在値を退避ステータスレジスタ（以下、「ＳＳＲ」と呼ぶ）１２８に保存してから書換える。ＳＲ１２５の書換え内容については後述する。この結果、割込ア

ドレスレジスタ１３０に登録されたアドレスに存在する、割込処理プログラムが起動する。割込処理プログラムは割込要因レジスタ１３０に記録された割込要因を参照して、これに対応する処理を実行する。

【００４５】なお、使用するプロセッサ１０１の種類により、割込処理プログラムのアドレスをメモリ上に格納し、当該格納位置のアドレスをプロセッサ内のレジスタに設定することもできる。

【００４６】また、使用するプロセッサ１０１の種類により、割込要因レジスタ１３０に割込要因を示す値を設定する代りに、割込要因ごとに異なる割込処理プログラムのアドレスを使用することもできる。この場合、各割込要因ごとの割込処理プログラムで要因を示す値を特定の汎用レジスタに記録してから共通の割込処理プログラムにジャンプし、当該共通割込処理プログラムで割込要因レジスタ１３０に記録された要因を参照する代りに前記の汎用レジスタを参照するようにすれば、図２で示したプロセッサと全く同様の割込処理を行うことができる。

【００４７】ＣＰＵ１２１、キャッシュメモリ１２２、各レジスタ１２３～１２５，１２７～１３０は、互いに、データ転送を行うデータバス１３１、アドレス指定を行うアドレスバス１３２によって接続されている。

【００４８】図３は、プロセッサ１０１が全割込マスク機能と割込レベルマスク機能とを装備している場合の、ＳＲ１２５の構成例である。ＳＲ１２５は、割込ブロックビット１４１と割込マスクレベルフィールド１４２を有する。割込ブロックビット１４１がＯＮの場合、プロセッサ１０１に対する全ての割込がマスクされる。割込マスクレベルフィールド１４２は現在の割込マスクレベル値を示し、これ以下の割込レベルの割込は受け付けられない。

【００４９】図３の例では、割込マスクレベルフィールド１４２は４ビット長であり、合計１６種類のマスクレベルを指定可能である。通常、割込レベル０は「割込が発生していない」ことを意味し、割込マスクレベルを０に設定すると「割込マスクを行わない」という指定になるため、実質的には１５種類となる。

【００５０】プロセッサ１０１の割込マスクレベルフィールド１４２のビット数を変更することにより、使用できる割込レベルの種類を増減させることができる。特権モードビット１４９がＯＮの場合、のプログラムは特権モードで動作しており、プロセッサ１０１に対する全ての操作が可能である。一方、特権モードビット１４９がＯＦＦの場合、動作中のプログラムは非特権モードで動作し、一部の機能、例えば、制御用レジスタやメモリ管理レジスタへのアクセスなどが禁止される。

【００５１】一般に、ＯＳは特権モードで動作する。一方、ＯＳ上で動作するタスクは、ＯＳの種類により、特権モードで動作するものや非特権モードで動作するものがある。

【００５２】図４は、プロセッサ１０１が全割込マスク機能と個別割込マスク機能とを装備している場合の、ＳＲ１２５の構成例である。ＳＲ１２５は実際には二つのレジスタ（実行状態レジスタ１４３と割込マスクレジスタ１４４）から構成され、ＳＳＲ１２７もこれら二つのレジスタを格納する領域を備える。図３と同様、実行状態レジスタ１４３内に割込ブロックビット１４１が設けられている。割込マスクレジスタ１４４内の割込マスクビット１４５～１４８はそれぞれ別々の割込に対応しており、割込マスクビットのいずれかをＯＮとした場合、当該ビットに対応する割込が受け付けられなくなる。

【００５３】図３に示すＳＲの例は、図４のＳＲの特殊例とみなせる。たとえば、割込マスクビット１４５のみＯＮとなっている状態をレベル１とし、割込マスクビット１４５、１４６の二つがＯＮとなっている状態をレベル２、割込マスクビット１４５～１４７の三つがＯＮとなっている状態をレベル３、…、のように対応させることができる。このため、以降、ステータスレジスタ１２５は図４に示す構成を有するとして説明する。

【００５４】割込を受理した場合、割込コントローラ１２６によりＳＲ１２５が書換えられる。一般的なプロセッサでは、割込ブロックビット１４１および特権モードビット１４９がＯＮに書換えられる。

【００５５】ＯＳ、デバイスドライバ、および特権モードで動作するタスクは、割込ブロックビット１４１や割込マスクレジスタ１４４の内容を書き換えることが可能で、割込マスクレベルを設定できる。これにより、割込処理実行中に同一割込やより優先度の低い割込が発生することを禁止したり、あるいは排他制御を実現することができる。

【００５６】以下、その動作について詳細に説明する。

【００５７】本実施例では、動作している複数のＯＳを全て同等のものとして扱っている。従って、以下の説明で一つのＯＳ上での動作、または一つのＯＳから他の一つのＯＳへの動作しか説明しない部分があるが、これらの処理を他のＯＳに対しても同様に適用することが可能である。

【００５８】図５に詳細なソフトウェアの機能ブロック図を示す。ただし、一部のコンポーネントについては図５に図示しておらず、図６～図８に示している。図５～図８に記載されている全てのコンポーネントはメモリ１０２上に格納されており、プロセッサ１０１にて実行される。以下の説明では、計算機システム上でのいくつかの動作パターンに分割して、各コンポーネントの詳細な動作を説明する。なお、図９以降の各図面が各コンポーネントの処理フローならびに処理で使用するデータ構造を示している。

【００５９】第一に、各ＯＳ上のタスクが当該ＯＳのスケジューリング機能により切替えながら動作している場

合の動作について説明する。

【００６０】第一ＯＳ３０１内には、タスクに対してサービスを提供するシステムコールプログラム３０８、およびタスク切替を行うスケジューラ３０９を有する。また、第一ＯＳ３０１上で動作するタスクがＯＳ切替部２０１の処理を利用するための関数を格納したＯＳ切替部利用ライブラリ３１０を有する。

【００６１】図５では、ＯＳ切替部ライブラリ３１０をタスクＡ３０２が利用している構成を示しているが、他のタスクからも利用可能である。タスクＡ３０２は、ＯＳ切替部ライブラリ３１０が提供する関数を呼出すことで処理を依頼する。呼出されたＯＳ切替部ライブラリ３１０内の関数は、タスクＡ３０２の一部として動作し、ＯＳ切替部２０１の処理関数を呼出したり、必要なシステムコール３０８を呼出したりして、依頼された処理を実現する。

【００６２】なお、タスクが専用の論理メモリ空間で動作している、あるいはタスクが非特権モードで動作しているなどの理由により、タスクの一部として動作するＯＳ切替部ライブラリ３１０が直接ＯＳ切替部２０１の関数の呼出すことができない場合、ＯＳ切替部利用ライブラリの機能の一部を、デバイスドライバなどの形でＯＳ内に置けばよい。

【００６３】スケジューラ３０９は、第一ＯＳ３０１上に存在する各々のタスクに対応してタスク管理ブロック３２１をメモリ上に確保し、各タスクを管理する。

【００６４】タスク管理ブロックの内部構造例を図９に、タスク管理ブロックの管理構造例を図１０に示す。タスク管理ブロック３２１の内部には、複数のタスク管理ブロックをキューに接続して管理するためのキュー接続用ポインタ３２１１、当該タスクの状態を表す状態フラグ３２１２、当該タスクが中断した時点のＰＣ、ＳＲ、汎用レジスタ値等を保存するためのレジスタ退避領域３２１３を含む。また、その他に中断理由や優先順位等の情報を含む場合もある。

【００６５】スケジューラ３０９は、実行可能なタスクに対応するタスク管理ブロックを、優先順位毎の実行可能タスクキュー（３２２、３２３など）に接続して管理する。従って、実行可能タスクキューは、優先順位の数だけ存在することになる。ただし、優先度の高いタスクのタスク管理ブロックをより前方に接続するという規則を追加することにより、単一の実行可能タスクキューにより管理することも可能である。

【００６６】入出力装置１０５に対する処理完了待ち、あるいはミューテックス・セマフォ・イベントなど第一ＯＳ３０１内の同期処理メカニズムによる再開待ちといった理由で処理を中断しているタスクに対応するタスク管理ブロックを、実行中断タスクキュー３２４に接続して管理する。なお、実行中断理由ごとに異なるキューで管理することもある。これらの各キューにタスク管理ブロックを接続するためにキュー接続用ポインタ３２１１が使用される。

【００６７】スケジューラ３０９は、システムコール３０８の処理や後述する割込ハンドラ３０７の処理が終了した時点で、現在実行中のタスクが実行可能でなくなった場合または現在実行中のタスクより高い優先順位のタスクが実行可能となった場合に、実行タスクを切替えるために呼出される。スケジューラ３０９を起動する可能性のあるシステムコールには、（ア）タスクの生成・終了・停止・再開（たとえば、自分より優先順位の高いタスクを生成した場合）、（イ）排他制御の実行・終了（たとえば、排他制御待ち状態に移行した場合）（ウ）タスクの優先順位変更（たとえば、自己の優先順位を低下させた場合）といったものがある。

【００６８】スケジューラ３０９の処理フローを図１１に示す。スケジューラ３０９は、実行中だったタスクのレジスタを当該タスクのタスク管理ブロック内にあるレジスタ退避領域３２１３に保存し（処理４０１）、実行可能タスクキューを優先順位が高い方から順に検索する（処理４０２）。

【００６９】タスク管理ブロックが一つも無かった場合は、処理４０３で判定され、ＯＳ間優先順位管理モジュール２１１にアイドル状態であることを通知する（処理４０５）。そして実行可能タスクが存在しないのであるから、アイドルループへ移行し（処理４０８）、実行すべきタスクが現れるまで何も行わない。

【００７０】一方、タスク管理ブロックが存在した場合には、当該タスクの優先順位をＯＳ間優先順位管理モジュール２１１に通知する（処理４０４）。ただし、通知する優先順位は全てのＯＳに渡って統一的に決められた優先順位を使用するものとし、第一ＯＳ３０１内で管理されている優先順位から予め決められた対応関係により変換して通知する。

【００７１】ＯＳ間優先順位管理モジュール２１１では、ここで（ア）通知した優先順位と、（イ）他のＯＳで当該ＯＳが中断する前に動作していたタスクの優先順位とを比較する。（以下、（ア）（イ）をまとめて、単に「当該ＯＳの優先順位」と呼ぶ。）通知した優先順位が、第二ＯＳ３５１の優先順位よりも高いか同じだった場合、処理４０４はすぐにリターンする。

【００７２】一方、通知した優先順位が第二ＯＳ３５１の優先順位よりも低かった場合、この時点で第一ＯＳ３０１の実行が中断され第二ＯＳ３５１に切替わり、その後第二ＯＳ３５１で動作しているタスクの優先順位が低下して第一ＯＳ３０１の優先順位の方が高くなった時点で処理４０４が終了し戻ってくる。ここで選択したタスク管理ブロックをキューから取出し（処理４０６）、レジスタ退避領域３２１３の内容をプロセッサ１０１の各レジスタに復旧し（処理４０７）、選択したタスクの処理が再開される。なお、当該タスクが新規作成された場

合はレジスタ退避領域３２１３には初期レジスタ値が登録されており、同じ処理でタスクの新規起動が行われる。

【００７３】ＯＳ間優先順位管理モジュール２１１の処理フローを図１２に示す。ＯＳ間優先順位管理モジュール２１１は、各ＯＳが最後に通知した優先順位を記録する変数を持つ。この変数が保持する値は、前述の当該ＯＳの優先順位に相当するため、以下、この変数を「ＯＳ優先順位変数」と呼ぶ。ＯＳのスケジューラ３０９ないし３５９から優先順位の通知を受けると、当該ＯＳに対応するＯＳ優先順位変数の内容を通知された値に更新し（処理４１１）、全てのＯＳ優先順位変数を比較してもっとも高い優先順位を持つＯＳを次に動作するＯＳとして選択する（処理４１２）。

【００７４】なお、複数のＯＳ優先順位変数が同一の最高値を持つ状態になった場合、当該ＯＳ優先順位変数に対応するＯＳのうち任意のものを選択すればよい。また、これらの複数のＯＳの中に最後に優先順位を通知したＯＳが含まれる場合には、切替オーバヘッドを削減するために当該ＯＳを選択するものとする。

【００７５】次に動作するＯＳを選択した後、遅延要求管理モジュール２０３を呼出す（処理４１３）。遅延要求が存在する場合の処理は後述するので、ここでは遅延要求が存在しなかったものとして説明を続ける。処理４１２で選択したＯＳが最後に動作していたＯＳだった場合、すわなち最後に優先順位を通知したＯＳの優先順位が最も高かった場合には、処理４１３がリターンしてくるので、呼出元であるスケジューラ３０９ないし３５９にそのままリターンする（処理４１４）。この結果、当該ＯＳのスケジューラが再開して選択したタスクが実行される。

【００７６】一方、処理４１２で選択したＯＳが最後に動作していたＯＳと異なる場合、遅延要求管理モジュール２０３はＯＳコンテキスト切替モジュール２０２を呼出して動作ＯＳを切替える。処理４１３はリターンしない。なお、いずれかのＯＳのスケジューラ３０９、３５９がＯＳ優先順位管理２１１の呼出を行わない構成とした場合、当該ＯＳに対応するＯＳ優先順位変数を固定値とすることで、当該ＯＳ上の全てのタスクが前記固定値の優先順位で動作しているものとみなしてＯＳ間の優先順位を管理することが可能である。

【００７７】ＯＳコンテキスト切替モジュール２０２の処理フローを図１３に示す。ＯＳコンテキスト切替モジュール２０２は、ＯＳ切替部２０１内の他のコンポーネントから呼出され、動作するＯＳを切替え、必要に応じて割込と同じ状態を作り出す。ＯＳコンテキスト切替モジュール２０２はメモリ１０２上に各ＯＳごとのコンテキスト退避領域を確保し、管理している。ここでは、ＯＳ間優先順位管理モジュール２１１からのＯＳ切替要求を処理する部分についてのみ説明する。

【００７８】図１３の処理４２１ならびに処理４２２はいずれもＮＯと判定され、プロセッサ１０１の各レジスタの現在値を直前に動作していたＯＳ（以下、「切替元ＯＳ」と呼ぶ）のコンテキスト退避領域に保存し（処理４２５）、これから動作させようとしているＯＳ（以下、「切替先ＯＳ」と呼ぶ）のコンテキスト退避領域にある値をプロセッサ１０１の各レジスタに復旧する（処理４２６）。この後ＰＣも保存されていた値に復旧し（処理４２８）、切替先ＯＳの中断点から動作が再開する。

【００７９】処理４２５をより詳細に説明する。ＯＳ切替部２０１に入ってからＯＳ切替部２０１内の各モジュールで変更したレジスタについては、現在値ではなくＯＳ切替部２０１に入った時点の値を保存する。これにより、当該ＯＳの動作が再開された時点において、ＯＳ切替部２０１に切替る前の状態を完全に復旧できる。

【００８０】また、ＯＳ切替部２０１内のモジュールを関数呼出の形で明示的に呼出した結果で処理４２５を実行する場合には、ＰＣの値として当該呼出関数からのリターンアドレスを保存し、リターン値を示すレジスタの値として当該呼出関数のリターン値を保存し、その他のレジスタについては前述のとおりＯＳ切替部２０１に入った時点の値を保存することがある。これにより当該ＯＳの動作が再開された時点において当該呼出関数がリターンした状態から実行されることになる。

【００８１】以上で説明した動作により、各ＯＳ上のタスクが、全てのＯＳに渡って統一的に決められた優先順位に従って、順次ＯＳを切替えられながら動作していくことになる。

【００８２】第二に、割込が入った場合の動作について説明する。ここで説明する割込とは、前述の外部割込や内部割込であり、遅延要求割込については後述する。

【００８３】割込を受付けた場合の割込処理プログラムとして、図５に示されている共通割込ハンドラ２１２が呼出される。すなわち、割込アドレスレジスタ１２９には共通割込ハンドラ２１２のアドレスを設定している。割込要因ごとに第一ＯＳ３０１または第二ＯＳ３５１のいずれかで処理するものとして割当を行う。共通割込ハンドラ２１２がその割当に基づき、ＯＳコンテキスト切替モジュール２０２に当該ＯＳに対する割込状態の発生を依頼する。

【００８４】ＯＳコンテキスト切替モジュール２０２は、当該ＯＳの割込ハンドラ３０７ないし３５７を呼出し、割込処理を実行させる。各割込ハンドラは必要に応じてシステムコールを発行し、その結果スケジューラが実行されてタスクの切替が発生する場合もある。例えば、入出力装置１０５の処理完了により割込が発生し、当該処理の完了を待機していたタスクの中断が解除されて当該タスクが再開される場合である。また別な例として、タイマ装置１０８からの割込発生により当該ＯＳの

内部時間が更新され、ラウンドロビン方式のスケジューリングを行うために現在実行中のタスクが中断され他の同一優先順位の実行待ちタスクにタスクスイッチされる場合がある。なお、本計算機システムでは、一部の割込要因をＯＳ切替部２０１の内部コンポーネントで処理するよう定義することもできる。

【００８５】割込を受付けた場合には、即座に処理を行うＯＳ上の割込ハンドラ３０５ないし３５５、あるいはＯＳ切替部２０１の内部コンポーネントを呼出す。従って、ある処理の実行中に、当該実行中の処理よりも処理の必要性が低い割込が発生して、その割込ハンドラにより当該実行中の処理が中断されることを防止するためには、割込マスクレベル値を引上げることが必要である。

【００８６】共通割込ハンドラ２１２の処理フローを図１４に示す。共通割込ハンドラ２１２は、図１５に示す割込割当テーブル２３１を持っており、割込要因レジスタ１３１の値と、当該割込要因レジスタ２３１の値に対応する割込を処理すべきＯＳとの組合わせを記録している。図１５に示した例の場合、割込要因レジスタ１３１に設定される値は、２００（１６進数）から始り、２０（１６進数）刻みの値であるものとしている。また、以下の説明では、２００（１６進数）はタイマ装置１「１０８１」からの割込、２２０（１６進数）はタイマ装置２「１０８２」からの割込、２４０（１６進数）はタイマ装置３「１０８３」からの割込を受付けた場合に設定される値とする。ここで、割込割当テーブル２３１の内容は固定されたものとしているが、各ＯＳや各ＯＳ上で動作するタスクからの要求により内容を動的に変更することも可能である。

【００８７】共通割込ハンドラ２１２は、割込発生時点で呼出され、割込禁止で動作する。まず、プロセッサ１０１の割込要因レジスタ１３１を参照し、割込要因を取得する（処理４４１）。次に割込割当テーブル２３１から当該割込を処理するＯＳを取得する（処理４４２）。ここで割当がＯＳ切替部２０１となっていた場合（処理４４３）、ＯＳ切替部２０１内の当該割込を処理するモジュールを呼出して処理を実行し（処理４４５）、終了後に当該処理で使用したレジスタを元に戻してからプロセッサの割込リターン命令（ＲｅＴｕｒｎ　ｆｒｏｍ　Ｅｘｃｅｐｔｉｏｎ。以下、「ＲＴＥ」と呼ぶ）により割込発生時点の動作に復帰する（処理４４６）。

【００８８】ここで割込要因とＯＳ切替部２０１内の処理を担当するモジュール（呼出される関数）との関係は、別途割当表が存在することを想定しているが、割込割当テーブル２３１の一部として記録することも可能である。一方、割当がＯＳ切替部２０１ではない場合、当該ＯＳに対して割込発生状態とするよう、ＯＳコンテキスト切替モジュール２０２へ依頼する（処理４４４）。

【００８９】ＯＳコンテキスト切替モジュール２０２では、図１３に示すように、処理で使用したワークレジスタの値を、割込が発生した時点の内容に回復する（処理４２３）。ここで、割込発生時点で動作していたＯＳと当該ＯＳが異なっていた場合（処理４２４）、既に説明したように切替元ＯＳのレジスタを退避し（処理４２５）、切替先ＯＳのレジスタを復旧する（処理４２６）。続いて、当該ＯＳの割込ハンドラ３０７ないし３５７のアドレスをＰＣに設定することで、当該割込ハンドラの実行を開始する（処理４２９）。

【００９０】図１５に示した割込割当テーブル２３１の内容の場合、タイマ装置１「１０８１」からの割込が発生した場合は第一ＯＳ３０１に割込を発生させ、タイマ装置２「１０８２」からの割込が発生した場合は第二ＯＳ３５１に割込を発生させる。また、タイマ装置３「１０８３」からの割込が発生した場合はＯＳ切替部２０１内部のＯＳ状態管理モジュール２０７が呼出されるものとする。ＯＳ状態管理モジュール２０７で行う割込処理の内容は後述する。

【００９１】なお、本実施例ではタイマ装置が三つ存在するものとして、各タイマの割込に対応して各ＯＳないしＯＳ切替部２０１が呼出されるものとしたが、タイマ装置が一つしか無い場合でも、当該タイマ装置からの割込が発生するごとに予め決められた順序で各ＯＳないしＯＳ切替部２０１のいずれかへの割込として扱うことで同じ効果を得ることができる。

【００９２】なお、割込要因レジスタ１３１に、割込割当テーブル２３１に存在しない値が設定されていた場合があり得る。また、割込割当テーブル２３１の割当ＯＳの欄に第四の値として「無効」と設定されている場合があり得る。これらの場合、共通割込ハンドラ２１２で単純にＲＴＥ（処理４４６）を行い無視するか、またはハードウェアの重大エラーとして扱い、エラー表示、エラー記録、ないし計算機システムの停止などの処理を行うかの、いずれかとする。

【００９３】第一ＯＳ３０１の割込ハンドラ３０７の処理フローを図１６に示す。割込ハンドラ３０７も割込禁止の状態で呼出され、まず現時点のレジスタの内容を割込スタックに退避する（処理４５１）。割込要因レジスタ１３１から割込要因を取得し（処理４５２）、当該割込とレベルが同じかよりレベルの低い割込が発生しないように割込マスクレベルを設定してから割込禁止を解除する（処理４５３）。処理４５３以降では、より割込レベルの高い割込を受付けて、さらに割込ハンドラが呼出されて処理すること（これを「多重割込」と呼ぶ）が可能になる。そして、割込要因に対応する各々の処理を実行する（処理４５４）。

【００９４】一般的なＯＳの場合、処理４５４の部分についてはデバイスドライバなどの形でユーザの任意の処理を組込めるような仕組となっており、処理の組込み方法やシステムコール３０８ないし３５８の使用方法等が明確に規定されている。またＯＳによっては、処理４５

４の内容を動作中に動的に変更することも可能である。各処理が終ると再度割込禁止とし（処理４５５）、多重割込の処理中であった場合（処理４５６）や、割込処理の結果スケジューリングが発生しなかった場合（処理４５７）は、処理４５１で割込スタック上に退避したレジスタを復旧して（処理４６０）、ＲＴＥ命令により割込発生時点の処理に復帰する（処理４６１）。　一方、割込処理内でのシステムコール発行などによりスケジューリングが必要となった場合には、処理４５１で割込スタック上に退避したレジスタをそれまで動作していたタスクのタスク管理ブロック内のレジスタ退避領域３２１３に保存し、割込スタックを破棄する（処理４５８）。そしてスケジューラ３０９へジャンプして割込ハンドラ３０７の処理を終了する。

【００９５】以上で説明した動作により、割込を受付けた場合に、必要に応じてＯＳを切替えながら、当該割込を割込ハンドラにて処理される。

【００９６】第三に、ＯＳ間通信、特にＯＳ間メッセージ通信機能の動作について説明する。

【００９７】図１７にＯＳ間メッセージ通信の概要を示す。ＯＳ間メッセージ通信は、ＯＳ間通信モジュール２０６の一機能として提供される。ＯＳ間通信モジュール２０６内に、メッセージキューと呼ばれるバッファ領域が存在する。メッセージキューには、ＩＤ番号が付けられ各々を識別している。

【００９８】図１７には、ＩＤ＝１のメッセージキュー２４１と、ＩＤ＝２のメッセージキュー２４２の二つが示されている。図１７では、タスクＰ３５２が、ＯＳ切替部ライブラリ３６０を通じて、メッセージ２４３を送信している。メッセージの内容はメッセージキュー２４１に一旦コピーされ、その後、タスクＡ３０２がＯＳ切替部ライブラリ３１０を通じて当該メッセージを取出す（受信）。

【００９９】図１７のメッセージ２４４、２４５のように、メッセージキューは受信要求があるまでの間複数のメッセージを保持することが可能である。この場合、メッセージ受信一回につき、ＦＩＦＯ（Ｆｉｒｓｔ　Ｉｎ　Ｆｉｒｓｔ　Ｏｕｔ）方式で一つのメッセージのみを取出して返す。

【０１００】ところで、タスクＡ３０２のメッセージ受信が、タスクＰ３５２のメッセージ送信よりも先に行われた場合、メッセージキュー２４１には取出すべきメッセージが存在しない、すなわち空の状態である。この場合、タスクＡ３０２からの指定により二つの動作が選択可能である。第一はメッセージが存在しないものとしてエラーとする動作である。第二はメッセージが送信されるまでタスクＡ３０２の実行を停止し、メッセージが送信された時点で、Ａ３０２の実行を再開して当該メッセージを取出すという動作である。第二の動作を、以下、「受信待機」と呼ぶ。

【０１０１】以上のような手順で異なるＯＳ上のタスク間でのメッセージ通信を実現する。なお、ここで説明しているＯＳ間メッセージ通信の機能を使い、同一のＯＳ上で動作するタスク間でのメッセージ通信を行うことも可能である。

【０１０２】ＯＳ切替部ライブラリ３１０のメッセージ受信処理（処理４７０）の処理フローを図１８に示す。以下の説明では、メッセージキューが空であった場合に受信待機をするよう指定された場合についてのみ説明する。この処理では、第一ＯＳ３０１の同期機能の一つであるイベントを使う。ここで、各イベントはイベントを識別するＩＤ（以下、「イベントＩＤ」と呼ぶ）を持つものとする。各イベントはシグナル状態と非シグナル状態の二つの状態が存在し、これらの状態を変更するシステムコールが提供されているものとする。また、任意のタスクは任意のイベントについてシグナル状態になるのを待機することができ、待機している間ポーリングなどのＣＰＵ処理を一切行うことなく動作が中断しているものとする。

【０１０３】メッセージ受信処理４７０では、まずＯＳ間通信モジュール２０６に対して、メッセージキューのＩＤ番号と自分自身のタスクＩＤを指定して受信要求を行う（処理４７１）。ここでメッセージキューが空でなければ（処理４７２）、取出したメッセージの内容とともに呼出元にリターンする（処理４７３）。ＯＳ間通信モジュール２０６から何らかのエラーが返された場合も同様である。

【０１０４】一方、メッセージキューが空であった場合、ＯＳ間通信モジュール２０６は処理保留のリターン値を返してくるので、イベントを非シグナル状態で作成し、このイベントＩＤと自分自身のタスクＩＤを遅延要求ハンドラ３０５に登録した上で、イベントがシグナル状態になるまで待機する（処理４７４）。

【０１０５】当該イベントはメッセージが送信された場合ないし何らかのエラー状態が発生した場合に、遅延要求ハンドラ３０５によりシグナル状態に変更される。当該イベントがシグナル状態になった時点で、再度ＯＳ間通信モジュール２０６にメッセージ受信要求を行い、当該メッセージを取出し（処理４７６）、使用したイベントを削除した上で呼出元にリターンする（処理４７７）。処理４７６でエラーが返された場合も同様である。なお、処理４７６では、当該要求が受信待機していた後の受信要求再発行であることを示す「再要求フラグ」を付ける。

【０１０６】ＯＳ間通信モジュール２０６では、当該フラグにより受信待機していた処理を、当該ＩＤのメッセージキューに対する他の受信要求と区別し、受信待機していた処理にメッセージを取出させるようにする。処理４７４のイベントの待機処理では、タイムアウトを設定する。もし、タイムアウトが発生した場合（処理４７

5）、ＯＳ間通信モジュール２０６に対して、受信待機している要求が無くなったことを知らせる通知を行い（処理４７８）、使用したイベントを削除した上で呼出元には、タイムアウトエラーでリターンする（処理４７９）。

【０１０７】なお、以上の説明では、受信待機が発生する毎にイベントを生成、削除するように記したが、あらかじめタスク毎にイベントを作成したり、いくつかのイベントを作成しておき使用されていないものを順次使用するということも可能である。

【０１０８】ＯＳ間通信モジュール２０６のメッセージ受信処理（処理４８０）の処理フローを図１９に示す。ここでも、メッセージキューが空であった場合、受信待機をするよう指定された場合についてのみ説明する。

【０１０９】ＯＳ間通信モジュール２０６は、各メッセージキュー毎に、受信待機中フラグと受信遅延要求発行フラグという二つの変数を持っている。このうち受信待機中フラグは、当該メッセージキューが空であったためにメッセージの受信待機を行っているタスクが存在することを示すフラグで、受信待機を行っているタスクの動作ＯＳとタスクＩＤが格納されている。受信遅延要求発行フラグは、受信待機が行われた後にメッセージ送信され、受信待機しているタスクからの再度の受信要求により当該メッセージが取出されるのを待機している状態であることを示すフラグである。

【０１１０】まずメッセージ受信要求を受けると、当該メッセージキューの受信待機中フラグを調べ（処理４８１）、もし待機中ならばエラーリターンとする（処理４８２）。すなわち、同時に複数のタスクが受信待機をすることはできない。

【０１１１】次に、受信遅延要求発行フラグが設定されていない場合、当該キューにメッセージが存在するか確認し（処理４８４）、存在すればＦＩＦＯ式にメッセージを一つ取出し、当該メッセージを本処理の呼出元が用意したバッファにコピーしてリターンする（処理４８５）。一方、メッセージが存在しない場合、要求元タスクの動作ＯＳとタスクＩＤを当該キューの受信待機中フラグに設定し（処理４８６）、受信待機を受付けたことを示す「処理保留」のリターン値でリターンする（処理４８７）。

【０１１２】一方、受信遅延要求発行フラグが設定されている場合、受信待機が行われた後にメッセージ送信されたことを示している。当該受信待機を行っているタスク以外から当該キューに対するメッセージ受信要求があった場合、すなわち受信要求に「再要求フラグ」が付いていなかった場合（処理４８８）、当該要求はエラーリターンとする（処理４９０）。一方、当該受信待機を行っているタスクからの受信処理要求（図１８の処理４７６など）があった場合、受信遅延要求発行フラグをＯＦＦした上で（処理４８９）、前述のメッセージ取出し処理４８５を行う。

【０１１３】ＯＳ間通信モジュール２０６のメッセージ送信処理（処理５００）の処理フローを図２０に示す。

【０１１４】まず送信されたメッセージを指定のキューにコピーする（処理５０１）。次に当該キューの受信待機中フラグを調べ（処理５０２）、受信待機中でなければそのまま呼出元にリターンする（処理５０６）。一方、受信待機中フラグが設定されていた場合、当該フラグに記録されているタスクＩＤを取出した後当該フラグをクリアし（処理５０３）、当該キューの受信遅延要求割込フラグをＯＮにする（処理５０４）。そして、受信待機しているタスクの動作を再開させるよう、遅延要求管理モジュール２０３に、処理５０３で取出したタスクＩＤのタスクに対するタスク再開の遅延要求の追加を依頼する（処理５０５）。

【０１１５】遅延要求管理モジュール２０３は、遅延要求を蓄積してすぐにリターンする場合もあれば、ＯＳコンテキスト切替モジュール２０２を通じて遅延要求割込を発行し遅延要求割込ハンドラへ処理が移行する場合もある。後者の場合、本送信処理を要求したタスクの属するＯＳが当該タスクの動作を再開する時点で、処理５０５のリターン時点に復旧するようにＰＣの退避を行う。これによりいずれの場合でも、呼出元へのリターンが行われる（処理５０６）。

【０１１６】遅延要求管理モジュール２０３の処理フローを図２１～図２３に示す。

【０１１７】遅延要求管理モジュール２０３は、大きく分けて二つの状況で呼出される。第一は新たな遅延要求が発生した場合である。この場合、当該要求先ＯＳに対応した遅延要求キュー２０４ないし２０５に要求を一旦格納する（処理５１２）。第二は、ＯＳ切替部２０１内の各モジュールがＯＳコンテキスト切替モジュール２０２にＯＳの切替を要求する場合で、共通割込ハンドラ２１２からの割込発生要求を除き、遅延要求管理モジュール２０３を経由する形での要求とする。この場合処理５１２はスキップされる（処理５１１）。

【０１１８】遅延要求管理モジュール２０３は、図５に示すように、各ＯＳに対する遅延要求割込許可フラグ２０９ないし２１０を持っている。このフラグがＯＦＦの場合は、当該ＯＳに対して一旦遅延要求割込を発行して当該割込に対する遅延要求割込ハンドラの処理が完了していないことを示している。また、遅延要求管理モジュール２０３は、各ＯＳに対する遅延要求割込レベルを記録する変数を持っていて、予め当該ＯＳに対する遅延要求割込の割込レベルを設定しておく。

【０１１９】遅延要求管理モジュール２０３は、まず第一ＯＳの遅延要求割込レベルを現時点でのプロセッサ１０１の割込マスクレベルと比較し、割込マスクレベルの方が高ければ第一ＯＳに対する以降の処理をスキップする（処理５１３）。次に第一ＯＳに対する遅延要求割込

許可フラグ２０９を調べ、これがＯＦＦの場合は第一ＯＳに対する以降の処理をスキップする（処理５１４）。遅延要求割込ハンドラ３０５は第一ＯＳ遅延要求キュー２０４内にある遅延要求を全て取出すまで処理を継続するため、当該処理が完了するまでは新たに遅延要求の発生を通知する必要がない。

【０１２０】従って、無意味な遅延要求割込の発生およびそれに伴うＯＳ切替を抑止するために、処理５１４の判定を行っている。続いて、第一ＯＳ遅延要求キュー２０４に遅延要求が存在するか調べ（処理５１５）、要求が存在する場合には第一ＯＳ３０１に対する遅延要求割込許可フラグ２０９をＯＦＦにして（処理５１９）、遅延要求割込ハンドラ３０５に遅延要求の存在を通知するために遅延要求割込を発生させる（処理５２０）。

【０１２１】実際の遅延要求割込の発生はＯＳコンテキスト切替モジュール２０２にて行われる。遅延要求割込の発生要求があった場合は、図１３に示した処理フローの中の処理４２２で判定され、共通割込ハンドラからの割込発生要求と同様に扱われる。この際、処理４２３において、割込要因レジスタ１３０に対して予め決めておいた遅延要求割込を示す値を設定する。この値は、他の割込要因では設定されない値を使用する。第一ＯＳ３０１と第二ＯＳ３５１で、同一の値とすることも異なる値とすることも可能である。割込ハンドラ３０７では、割込要因レジスタ１３０に遅延要求割込を示す値が設定されていることを確認し、割込処理４５４として遅延要求割込ハンドラ３０５を起動する。

【０１２２】さて、遅延要求管理モジュール２０３の説明に戻る。第一ＯＳ３０１に対して遅延要求割込の発生を行わなかった場合、第二ＯＳ３５１に対しても同様に遅延要求割込を発生させるかどうかの判定を行う（処理５１６～処理５１８）。第二ＯＳ３５１に対して遅延要求割込を発生させると判断した場合は、第一ＯＳ３０１と同様に処理５１９および処理５２０を行う。

【０１２３】第一ＯＳ３０１および第二ＯＳ３５１のいずれに対しても遅延要求割込の発生を行わなかった場合、遅延要求管理モジュール２０３が呼出された理由を処理５１１と同じ判定で調べ、新たな遅延要求が発生した場合の呼出ならば呼出元にリターンし（処理５２２）、ＯＳコンテキスト切替モジュール２０２にＯＳの切替を要求する場合であればＯＳコンテキスト切替モジュール２０２にジャンプする（処理５２３）。

【０１２４】ここまでの説明では、遅延要求割込の発生判定を第一ＯＳ３０１、第二ＯＳ３５１の順で固定的に行うものとしたが、他の順序でも構わない。例えば、遅延要求割込の割込レベルが高い方を先に行ったり、遅延要求の追加時は追加された要求のＯＳを先に行ったりということも考えられる。

【０１２５】また、遅延要求割込許可フラグの状態、および遅延要求割込の割込レベルと割込マスクレベルの比較によって割込発生するかどうかを決めているが、他の基準を追加することも可能である。例えば、遅延要求割込を発生させることができるＯＳの動作優先順位を予め決めておき、当該ＯＳの優先順位の現在値が予め決められた優先順位よりも低いときに限り遅延要求割込を発生させるようにすることができる。

【０１２６】次に、第一ＯＳ３０１の遅延要求割込ハンドラ３０５の処理フローを図２４に示す。前述のとおり、遅延要求割込ハンドラ３０５は、割込ハンドラ３０７の中の割込処理４５４の一つとして動作する。割込ハンドラ３０７で説明したとおり、遅延要求割込ハンドラ３０５の処理中は、第一ＯＳ３０１への遅延要求割込の割込レベルよりも割込レベルの高い割込を受付けることが可能である。

【０１２７】遅延要求割込ハンドラ３０５は、遅延要求キュー２０４から遅延要求を一つ取出す（処理５３１）。遅延要求が残っていないことがわかった場合（処理５３２）、第一ＯＳ遅延要求割込許可フラグ２０９をＯＮにして（処理５３３）、処理を完了する。

【０１２８】遅延要求が取出せた場合、その要求の種類を判定し（処理５３４）、各々対応する処理を行う。ここまでで説明してきたＯＳ間メッセージ通信の場合、メッセージの送信により、受信待機していたタスクを再開する要求（以下、「タスク再開要求」と呼ぶ）が取出されることになる。この場合、タスク再開要求には再開すべきタスクのタスクＩＤがパラメータとして付随しているので、受信待機する際に記録されたタスクＩＤとイベントＩＤの組から当該タスクＩＤに対応するを再開するためのイベントＩＤを取得し、当該イベントをシグナル状態にするシステムコールを発行する（処理５３５）。

【０１２９】イベントがシグナル状態になると、当該タスクで実行中だったＯＳ切替部利用ライブラリ３１０のメッセージ受信処理（処理４７０）が再開され、送信されたメッセージの取出しが行われることになる。遅延要求割込ハンドラ３０５は、遅延要求キュー２０４の要求が無くなるまで、要求の取出し（処理５３１）を繰返す。

【０１３０】なお、ここで説明した遅延要求割込ハンドラ３０５の処理の大部分を、割込ハンドラではなく第一ＯＳ３０１上で動作するタスクとして構成することも可能である。すなわち、遅延要求割込ハンドラ３０５は、当該処理タスクに割込の発生を通知するだけで処理を終了する。当該処理タスクでは前記通知を受け次第、ここまでで説明してきた遅延要求割込ハンドラ３０５の処理を実行する。第一ＯＳ遅延要求割込許可フラグ２０９をＯＮにする処理５３３が実行されるまでは、再度遅延要求割込が発生することはないので、遅延要求割込ハンドラと当該処理タスクの間で競合管理を行う必要はない。

【０１３１】ここまでの説明では、遅延要求割込ハンドラ３０５が直接遅延要求キュー２０４や遅延要求割込許

可フラグ２０９を操作するようにしている。これらのキューやフラグには遅延要求管理モジュール２０３もアクセスするため、競合管理が必要となる。第一ＯＳ３０１上のコンポーネントである遅延要求割込ハンドラ３０５とＯＳ切替部２０１上のコンポーネントである遅延要求管理モジュール２０３との間で競合管理をするには、複雑な処理が必要になる。これを解決するには、遅延要求管理モジュール２０３に、遅延要求キュー２０４からの要求取出し処理や遅延要求割込許可フラグ２０９の操作処理を用意し、遅延要求割込ハンドラ３０５はこれらの処理を呼出す形とすればよい。

【０１３２】以上で説明した動作により、異なるＯＳ上で動作するタスク間でのメッセージ通信機能が実現される。

【０１３３】第四に、遅延要求を使用するその他の処理について説明する。既に述べたとおり、遅延要求割込ハンドラ３０５は当該ＯＳのシステムコールを発行できるため、ＯＳ切替部２０１から各ＯＳに対する様々な処理要求の実現手段として使用できる。

【０１３４】まず、ＯＳ間通信の他の処理での使用を説明する。ＯＳ間メッセージ通信のメッセージキューが満杯の場合、当該メッセージキューに対する送信要求をメッセージキューにメッセージを格納する空きが出来るまで待機させる処理を実装することができる。この場合、前述のメッセージ受信待機と同様にタスク再開要求の遅延要求を使用すれば実現できる。

【０１３５】ただし、送信待機の場合は受信待機と異なり、複数の要求を同時に待機させてもよい。そのようにした場合、サイズの大きなメッセージがメッセージキューより取出されたときに、複数のタスク再開要求を同時に発行する必要がある。この場合、遅延要求管理モジュール２０３へは全ての要求を一括して追加するように依頼し、遅延要求キューに格納することが可能である。

【０１３６】しかし、これら複数の要求が複数のＯＳに対するものでありなおかつこれらのＯＳがいずれも遅延要求割込を発行できる状態にあった場合でも、いずれか一つのＯＳにしか遅延要求割込は発行できず、他のＯＳへの割込発生は再度遅延要求管理モジュール２０３が呼出されるまで遅れることになる。

【０１３７】次に、ＯＳ間通信モジュール２０６の機能の一つとして、異なるＯＳで動作するタスク間で競合管理を行うためのＯＳ間セマフォを実装する場合について説明する。

【０１３８】ＯＳ間セマフォは、基本的にはＯＳ間メッセージ通信の送受信処理と同様にして実現できる。すなわち、セマフォの獲得が成功した場合には獲得処理が即座にリターンし、一方獲得待ちとなった場合には処理保留でリターンする。そして、セマフォの解放処理で、セマフォ獲得待機タスクがある場合には当該タスクのタスク再開遅延要求を行えばよい。

【０１３９】一方、ＯＳ間セマフォの追加的な機能として、優先順位の継承を実装する場合がある。これはＯＳ間セマフォの獲得待機をしているタスクの実行優先順位が、既に当該セマフォを獲得しているタスクの優先順位よりも高かった場合に、既に当該セマフォを獲得しているタスクの優先順位を一時的に獲得待機をしているタスクの実行優先順位まで引上げる機能である。このためには、ＯＳ間通信モジュール２０６から各ＯＳに対して、タスクの優先順位変更を依頼しなければならない。この依頼はは、タスクの優先順位変更という遅延要求を追加し、遅延要求割込ハンドラ３０５内で当該要求取出し時にタスクの優先順位変更システムコールを発行することで実現できる。

【０１４０】次に、ＯＳのシステム時刻変更時の動作について説明する。既に述べたとおり、各ＯＳは、ＲＴＣ１０７に対して直接時刻を読み書きする代りに、ＲＴＣ管理モジュール２０８にこれらの要求を行う。第二ＯＳ３５１で管理するシステム時刻が変更された場合、変更後の新しいシステム時刻をＲＴＣ１０７にも反映させるため、ＲＴＣ管理モジュール２０８に設定依頼が行われる。

【０１４１】この場合の処理フローが、図２５である。依頼が合った場合、実際にＲＴＣ１０７へ時刻を設定し（処理５５１）、遅延要求管理モジュール２０３にＯＳ１に対する「時刻変更要求」の遅延要求の発行を依頼する（処理５５２）。遅延要求割込ハンドラ３０５は、図２４に示したように、取出した遅延要求が「時刻変更要求」であった場合には、ＲＴＣ１０７の時刻を取得し（処理５３６）、システム時刻をこの値に設定するシステムコールを発行する（処理５３７）。

【０１４２】ただし、処理５３７の実行中に、当該システムコール発行の結果再度ＲＴＣ管理モジュール２０８のＲＴＣ設定処理が呼出されないように予め設定しておく必要がある。なお、ＲＴＣ時刻にシステム時刻を一致させるシステムコールがあれば、処理５３６ならびに処理５３７は、一括して当該システムコールに置換えることができる。また、より簡易な処理としては、処理５３６を実行する代りに、処理５２２において、「時刻変更要求」の遅延要求に付随するパラメータに設定すべき時刻のデータを付加しておく方法も可能である。

【０１４３】さらに、ＯＳ状態管理モジュール２０７からの要求も遅延要求として実現することが可能である。図２４の処理フローではその一例として、「シャットダウン要求」の遅延要求処理を含んでいる。遅延要求割込ハンドラ３０５は、取出した遅延要求が「シャットダウン要求」であった場合には、シャットダウンを開始するシステムコールを発行する（処理５３８）。シャットダウン要求の他に、あるＯＳの動作状態が変化した場合、たとえばＯＳが起動した場合、ＯＳがシャットダウンした場合、ＯＳが異常停止した場合など、これを通知する

ための遅延要求を他のＯＳに行うことができる。遅延要求割込ハンドラ３０５が、このようなＯＳの状態変化を通知する遅延要求を受取った場合、状態変化を取扱うタスクを起動する、状態変化を通知するイベントをシグナル状態に変更する、といった処理を行うことが可能である。

【０１４４】図５では、遅延要求キュー２０４、２０５は各ＯＳ３０１、３５１ごとに一つだけ存在するように示されているが、以上で説明したような各遅延要求の重要性のレベル毎にキューを用意して、対応するキューに分けて蓄積することが可能である。

【０１４５】このような遅延要求キューの構成例を図６に示す。図６の構成の場合、第一ＯＳ遅延要求キュー２０４は、レベル１遅延要求キュー２０４１、レベル２遅延要求キュー２０４２、レベル３遅延要求キュー２０４３の集合体として構成される。レベル１遅延要求キュー２０４１にはシャットダウン要求などのＯＳ状態管理モジュール２０７からの処理要求が、レベル２遅延要求キュー２０４２にはＲＴＣ管理モジュール２０８から時刻変更要求が、レベル３遅延要求キュー２０４３にはタスク再開要求などのＯＳ間通信モジュール２０６からの処理要求が蓄積される。

【０１４６】ここで、遅延要求割込ハンドラ３０５が、レベルの小さいキューから順に遅延要求を取出すことにより、各要求はレベル１、レベル２、レベル３の順で優先して処理されることになる。なお、遅延要求管理モジュール２０３の中に遅延要求の取出し処理を実装する場合には、当該処理の中でレベルの小さいキューから順に遅延要求を取出すことになる。図６では、遅延要求キューを三つのキューで構成されるとしたが、キューの数を変更し各キューに優先順位を付けることも可能である。また、各キュー毎に、遅延要求割込の割込レベルを異なるものとしたり、遅延要求割込を発生させることができるＯＳの動作優先順位を変更したりすることも可能である。

【０１４７】以上のように、ＯＳ切替部２０１から各ＯＳ３０１、３５１に対する様々な処理要求の実現手段として、遅延要求が使用される。しかし、処理内容によっては遅延要求が適さず、ＯＳ切替部２０１から呼出されるコールバックルーチンで処理する必要がある。

【０１４８】遅延要求とコールバックルーチンのいずれを使うかの判断基準の例を図３０に示す。図３０で示した判断基準の場合、処理によりスケジューリングが発生する可能性がある場合や処理時間が長く処理中にＯＳ切替や割込発生を許可したい場合に遅延要求を使用し、即座に実行が必要な処理や当該ＯＳの割込処理が正常に行われていない場合の処理、あるいは処理によりスケジューリングが発生しない短時間の処理にコールバックルーチンを使用する。例えば、ＯＳ状態管理モジュール２０７から当該ＯＳを強制停止させる処理の場合、当該ＯＳが正常に動作していない場合に使われる可能性があるため、遅延要求を使用せずにコールバックルーチンを使用する必要がある。ただし、長時間の処理をコールバックルーチンで処理することは可能であり、また処理結果でスケジューリングが発生しない短い処理を遅延要求で処理することも可能である。

【０１４９】以上で説明した動作により、遅延要求を使用する各処理が実現される。

【０１５０】第五に、ＯＳ状態管理モジュール２０７の各ＯＳに対する監視タイマ機能（Ｗａｔｃｈ　Ｄｏｇ　Ｔｉｍｅｒ。以下、「ＷＤＴ」と呼ぶ）の動作について説明する。

【０１５１】図７にＷＤＴ機能に関するソフトウェアの機能ブロック図を示す。

【０１５２】ＯＳ状態管理モジュール２０７は、第一ＯＳ３０１用のＷＤＴカウンタ２１７、および第二ＯＳ３５１用のＷＤＴカウンタ２１８を管理している。これらのカウンタ２１７、２１８は当該ＯＳ３０１、３５１に対するＷＤＴ機能を開始したときに０にクリアされる。また、タスクからの要求があった場合にも、当該タスクが動作しているＯＳ用のＷＤＴカウンタが０にクリアされる。タスクからの０クリア要求は、当該タスクが動作していることを通知する意味を持ち、以下、これを「生存通知」と呼ぶ。共通割込ハンドラ２１２は、一定周期で発生するタイマ装置１０８３からの割込発生を受付け、ＯＳ状態管理モジュール２０７を呼出す。

【０１５３】ＯＳ状態管理モジュール２０７のタイマ割込処理（処理５６０）の処理フローを図２６に示す。

【０１５４】処理では、まず、第一ＯＳ３０１および第二ＯＳ３５１用のＷＤＴカウンタに値を１だけ加算する（処理５６１、処理５６２）。そして第一ＯＳ用ＷＤＴカウンタ２１７が予め指定された第一ＯＳ３０１のタイムアウト時間に相当するカウント値よりも大きいと判断された場合（処理５６３）、第一ＯＳ３０１のコールバックルーチン３０６を呼出して第一ＯＳが管理する入出力装置からの割込発生を抑止させ（処理５６４）、以後ＯＳコンテキスト切替モジュール２０２の処理で第一ＯＳへ切替えることを禁止する（処理５６５）。第二ＯＳに対しても同様の処理を行う（処理５６６～５６８）。

【０１５５】ここで説明した処理により、ＯＳの動作の正常性を確認する機能が提供される。生存通知を行うタスクの優先順位を高く設定しておけば、ＯＳ本体や優先順位の非常に高いタスク（たとえば当該ＯＳ上の統括処理を行うタスク）が動作できなくなった場合に限り、当該ＯＳが不正常だとみなされる。

【０１５６】一方、生存通知を行うタスクの優先順位を低く設定しておけば、当該優先順位以上の優先順位を持つタスクのいずれか一つに対してでも処理時間の割当が行われなかった場合に、当該ＯＳの動作を不正常として検出することになる。生存通知を行うタスクの優先順位

は、当該ＯＳ上で動作する各タスクの重要性と、当該ＯＳの状態管理や当該ＯＳ上の統括処理を行うタスクの処理内容を勘案して決定することになる。

【０１５７】本実施例では、ＷＤＴで異常を検出した場合、当該ＯＳの動作を即座に停止するものとしている。

【０１５８】一方、異常を検出した場合に、まず正規のシャットダウン手順での停止を当該ＯＳに要求し、一定時間以内にシャットダウンの完了通知がなかった場合には、強制的に停止することも可能である。このようにするには、ＷＤＴタイムアウト時間を越えた場合の処理５６４ないし５６７の代りに当該ＯＳに対するシャットダウン要求の遅延要求を発行し、ＯＳへの切替禁止（処理５６５ないし処理５６８）をしない。その後のタイマ割込処理においても、当該ＯＳのＷＤＴカウンタ加算５６１ないし５６２を継続する。

【０１５９】ここで当該ＯＳからシャットダウン完了の通知があった場合はタイマの加算を行わないようにするが、当該通知がないまま加算を継続して第二のタイムアウト時間を越えた場合、当該ＯＳの割込マスクを行うコールバックルーチン呼出（図２６の処理５６４ないし処理５６７）、および当該ＯＳへの切替禁止（処理５６５ないし処理５６８）を行えばよい。なお、ここで説明した一定時間以内にシャットダウンの完了通知がなかった場合に当該ＯＳを強制停止するという手順については、ＷＤＴのタイムアウト時以外にも、各ＯＳに対するシャットダウン要求を行う全ての場合に共通に適用することが可能である。

【０１６０】さらに、異常を検出して当該ＯＳを停止させた後に、自動的に当該ＯＳの起動処理を呼出し、当該ＯＳを再起動する手順を追加することも可能である。この場合、当該ＯＳ上タスクの不良論理などによる異常検出・再起動の繰返しを防ぐために、ＯＳ状態管理モジュール２０７内に各ＯＳの再起動回数を記録する変数を用意して、再起動回数が指定回数以上になった場合は自動再起動を抑止する機能を追加することが可能である。

【０１６１】この再起動回数の記録は、指定時間ごとに、あるいは最後の再起動が行われてから指定時間が経過した後にクリアする機能を追加することが可能である。また、再起動を要求してから指定した時間内に再度異常を検出した場合に、正常な起動が出来ないものとみなして自動再起動を抑止する機能を追加することが可能である。なお、ここで説明した自動再起動の処理については、ＷＤＴのタイムアウト時以外に、各ＯＳの異常終了検出時にも共通に適用することが可能である。

【０１６２】以上で説明した処理により、当該計算機システムの信頼性を向上させるためのＯＳ動作管理モジュール、特にＷＤＴ機能の動作が実現される。

【０１６３】第六に、ＯＳ間リモート手続き呼出機能（Ｒｅｍｏｔｅ　ＰｒｏｃｅｄｕｒｅＣａｌｌ。以下「ＲＰＣ機能」と呼ぶ）の動作について説明する。

【０１６４】図８にＯＳ間ＲＰＣ機能に関するソフトウェアの構成図を示す。ＯＳ間ＲＰＣ機能を使用する第一ＯＳ３０１上のタスクはＲＰＣ利用ライブラリ３１１に用意された処理関数を呼出す。ＲＰＣ利用ライブラリ３１１は、ＯＳ間通信モジュール２０６のＯＳ間メッセージ通信機能を使用して、第二ＯＳ３５１上にあるＲＰＣサーバ３５４に処理を依頼する。このとき使用されるメッセージキュー２１９は第二ＯＳ３５１のＲＰＣサーバ３５４に対する処理要求を送るための専用のもので、そのメッセージキューＩＤはＲＰＣ利用ライブラリ３１１が予め知っているものとする。

【０１６５】ＲＰＣサーバ３５４は、実際にはＲＰＣ管理タスク３５４１とＲＰＣ実行タスク３５４２の組合わせで構成される。ＲＰＣ実行タスク３５４２は、要求された関数（手続き）を実行する。この関数は通常のタスク環境で実行されているので、システムコール３５８を発行することが可能である。また、要求関数として、システムコールそのものを指定することも可能である。ＲＰＣ管理タスク３５４１は要求された処理が完了した後、当該処理要求を行ったタスクＡ３０２専用のＲＰＣ結果通知メッセージキュー２２０を使用して、処理結果を通知する。

【０１６６】ＲＰＣ利用ライブラリ３１１の初期化処理（処理５８０）の処理フローを図２７に示す。この初期化処理は、各タスクがＲＰＣを初めて利用するときに一度だけ呼出される。この処理を呼出したタスクが、その後のＲＰＣ処理要求を行った場合に、処理結果を受取るためのＲＰＣ結果通知メッセージキュー２２０を作成する（処理５８１）。

【０１６７】ＲＰＣ利用ライブラリ３１１のＲＰＣ実行処理（処理５９０）の処理フローを図２８に示す。タスクは、実行する関数名とパラメータを渡して処理５９０を呼出す。処理５９０では、第二ＯＳ３５１のＲＰＣ要求メッセージキュー２１９に対して、要求元から指定されたままの関数名とパラメータ、および初期化処理５８０で作成したＲＰＣ結果通知メッセージキュー２２０のメッセージキューＩＤを、一つの要求メッセージとして送信する（処理５９１）。その後、ＲＰＣ結果通知メッセージキュー２２０に対して結果を通知するメッセージが送られてくるのを受信待機する（処理５９２）。

【０１６８】第二ＯＳ３５１のＲＰＣ管理タスク３５４１の処理フローを図２９に示す。管理タスクは常時ＲＰＣ要求メッセージキュー２１９の受信待機をしておりメッセージが到着次第これを取出す（処理６０１）。要求メッセージを取出すと、ＲＰＣ実行タスク３５４２を起動し（処理６０２）、要求された実行関数名およびパラメータをＲＰＣ実行タスク３５４２に渡して実行開始を指示する（処理６０３）。そしてＰＲＣ実行タスク３５４２からの実行完了メッセージが送られてくるのを待機する（処理６０４）。なお、ここでいうメッセージは、

第二ＯＳ３５１の機能として提供される同期機能あるいは通信機能のいずれかを使用して実現する。

【０１６９】ＰＲＣ実行タスク３５４２は、関数名から当該関数のアドレスを返す第二ＯＳ３５１のシステムコールを呼出すことにより要求された関数のアドレスを解決し、当該関数を実行する。実行が完了した場合、関数のリターン値を含む終了メッセージをＲＰＣ管理タスク３５４１に送る。ここで、要求された処理やパラメータの正当性チェックは行わないため、処理が完了しなかったり、ＰＲＣ実行タスク３５４２が停止してしまったりする場合がある。

【０１７０】第二ＯＳ３５１のＲＰＣ管理タスク３５４１は実行完了メッセージの待機処理６０４において、タイムアウトを設定しており、前記のようにＰＲＣ実行タスク３５４２が正常に終了しない場合にはタイムアウトとなる（処理６０５）。この場合は、ＰＲＣ実行タスク３５４２を強制的に停止し（処理６０８）、処理要求元タスクのＲＰＣ結果通知メッセージキュー２２０に対してタイムアウトエラーの結果メッセージを送信する。

【０１７１】一方、ＰＲＣ実行タスク３５４２が正常に処理を終了して、終了メッセージが送られてきた場合、ＰＲＣ実行タスク３５４２を停止し（処理６０６）、終了メッセージに含まれるリターン値を含む結果メッセージを、処理要求元タスクのＲＰＣ結果通知メッセージキュー２２０に対して送信する（処理６０７）。

【０１７２】ＲＰＣ利用ライブラリ３１１のＲＰＣ実行処理（処理５９０）は、結果通知メッセージの到着で動作を再開し、その結果とともに、処理呼出元にリターンする（処理５９３）。ここで、あるタスクがＲＰＣ実行処理５９０を呼出すと、この処理が終了するまで当該タスクは実行を中断することになる。

【０１７３】ＲＰＣ管理タスク３５４１は予め決められた処理を実行するだけであり要求された処理の内容にかかわらず異常終了することはなく、またＲＰＣ実行タスク３５４２に対するタイムアウトが設定されているため、必ず何らかの処理結果のメッセージが送られる。すなわち、処理５９０に相当する関数内で指定したＲＰＣ処理が完結するかタイムアウトするかのいずれかでリターンしてくることが期待でき、異なるＯＳ間で通信する処理や、通信に伴うエラー処理などを意識することなく、他のＯＳ上での処理が実行できる。なお、ここではタイムアウト時間をＲＰＣサーバが保持している固定の値としたが、個々のＲＰＣ処理要求毎にタイムアウト時間を指定するようにしてもよい。

【０１７４】ここで説明したＲＰＣサーバの構成では、ＲＰＣ実行タスク３５４２でエラーが発生して強制停止した場合にはＲＰＣ管理タスク３５４１内でのタイムアウト発生まで待つものとしたが、タスクの停止を通知する機能が当該ＯＳに備わっている場合には、この通知を検出して、即座にエラーの通知を行う構成も可能である。

【０１７５】また、ＲＰＣサーバの構成では、関数アドレスの解決はＲＰＣ実行タスク３５４２で行うものとしたが、これをＲＰＣ管理タスク３５４１で実行し、ＲＰＣ実行タスク３５４２には解決済のアドレスを渡す構成も可能である。さらに、一旦解決した関数アドレスをＲＰＣ管理タスク３５４１内に置いたテーブルに記録し、同一の関数を再度実行する場合には当該テーブルからアドレスを取得することによって、アドレスの解決処理のオーバヘッドを小さくできる。また、システムコールによる関数アドレスの解決の代りに、予めＲＰＣ管理タスク３５４１のビルド時にアドレスを解決して、前記記録テーブルに保持しておく方法も可能である。

【０１７６】ＲＰＣサーバは、ＲＰＣ実行タスク３５４２を一つとしたが、複数のＲＰＣ要求を同時に実行するために、ＲＰＣ実行タスク３５４２を複数同時に起動する構成も考えられる。ただし、この場合には、起動したＲＰＣ実行タスク３５４２全てに対して実行完了メッセージをタイムアウト付で待機することが必要となる。例えば、起動したＲＰＣ実行タスク３５４２と同数の管理サブタスクを別に起動し、各々一つのＲＰＣ実行タスク３５４２からの実行完了メッセージを待機するといった処理が必要になる。逆に、簡易な構成として、ＲＰＣ実行タスク３５４２の処理をＲＰＣ管理タスク３５４１の一部とする方法も可能である。この場合、要求関数の実行で例外が発生した場合に、当該ＯＳが提供するエラー回復手段（シグナル機構や構造化例外処理などの手段）を使用してタスクの動作を継続させ、エラー発生を処理要求元タスクのＲＰＣ結果通知メッセージキュー２２０に対して送信する構成にしなければならない。

【０１７７】以上で説明した処理により、異なるＯＳ上の処理を用意に呼出すためのＯＳ間ＲＰＣ機能の動作が実現される。

【０１７８】以上のようにして複数個のオペレーティングシステム間で処理要求を通信するのであるが、オペレーティングシステムが処理要求を割込み処理するようにしているので、オペレーティングシステムに通信用のハンドラ機能を一つ追加するだけでよくオペレーティングシステム間の処理要求の通信をオペレーティングシステムの簡単な改造で行える。

【０１７９】なお、上述の実施例においてはハード構成で説明しているが、プログラムの処理手順としてＣＤ－ＲＯＭなどの記憶媒体に収納し、コンピュータなどの演算処理装置を用いて本発明を実施できることは明らかなことである。

【０１８０】

【発明の効果】本発明によれば、オペレーティングシステムが処理要求を割込み処理するようにしているので、オペレーティングシステムに通信用のハンドラ機能を一つ追加するだけでよくオペレーティングシステム間の処

理要求の通信をオペレーティングシステムの簡単な改造で行える。

【図面の簡単な説明】

【図１】本発明の一実施例の構成を示す図である。

【図２】計算機システムのハードウェア構成を示す図である。

【図３】割込レベルマスク機能を装備する場合のステータスレジスタを示す図である。

【図４】個別割込マスク機能を装備する場合のステータスレジスタを示す図である。

【図５】ソフトウェア構成の詳細を示す機能ブロック図である。

【図６】遅延要求キューの構成例を示す図である。

【図７】ソフトウェア構成の詳細を示す機能ブロック図である。

【図８】ソフトウェア構成の詳細を示す機能ブロック図である。

【図９】タスク管理ブロックの内部構造の例を示す図である。

【図１０】タスク管理ブロックの管理例を示す図である。

【図１１】スケジューラの処理フローを示す図である。

【図１２】ＯＳ間優先順位管理モジュールの処理フローを示す図である。

【図１３】ＯＳコンテキスト切替モジュールの処理フローを示す図である。

【図１４】共通割込ハンドラの処理フローを示す図である。

【図１５】割込割当テーブルの構造の例を示す図である。

【図１６】割込ハンドラの処理フローを示す図である。

【図１７】ＯＳ間メッセージ通信機能の動作を示す図である。

【図１８】ＯＳ切替部利用ライブラリのメッセージ受信処理の処理フローを示す図である。

【図１９】ＯＳ間通信モジュールのメッセージ受信処理の処理フローを示す図である。

【図２０】ＯＳ間通信モジュールのメッセージ送信処理の処理フローを示す図である。

【図２１】遅延要求管理モジュールの処理フローを示す第一の図である。

【図２２】遅延要求管理モジュールの処理フローを示す第二の図である。

【図２３】遅延要求管理モジュールの処理フローを示す第三の図である。

【図２４】遅延要求割込ハンドラの処理フローを示す図である。

【図２５】ＲＴＣ管理モジュールのＲＴＣ設定処理の処理フローを示す図である。

【図２６】ＯＳ状態管理モジュールのタイマ割込処理の処理フローを示す図である。

【図２７】ＲＰＣ利用ライブラリの初期化処理の処理フローを示す図である。

【図２８】ＲＰＣ利用ライブラリのＲＰＣ実行処理の処理フローを示す図である。

【図２９】ＲＰＣ管理タスクの処理フローを示す図である。

【図３０】遅延要求とコールバックを選択する基準の例を示す図である。

【図３１】ＯＳ間通信機能の従来技術を示す構成図である

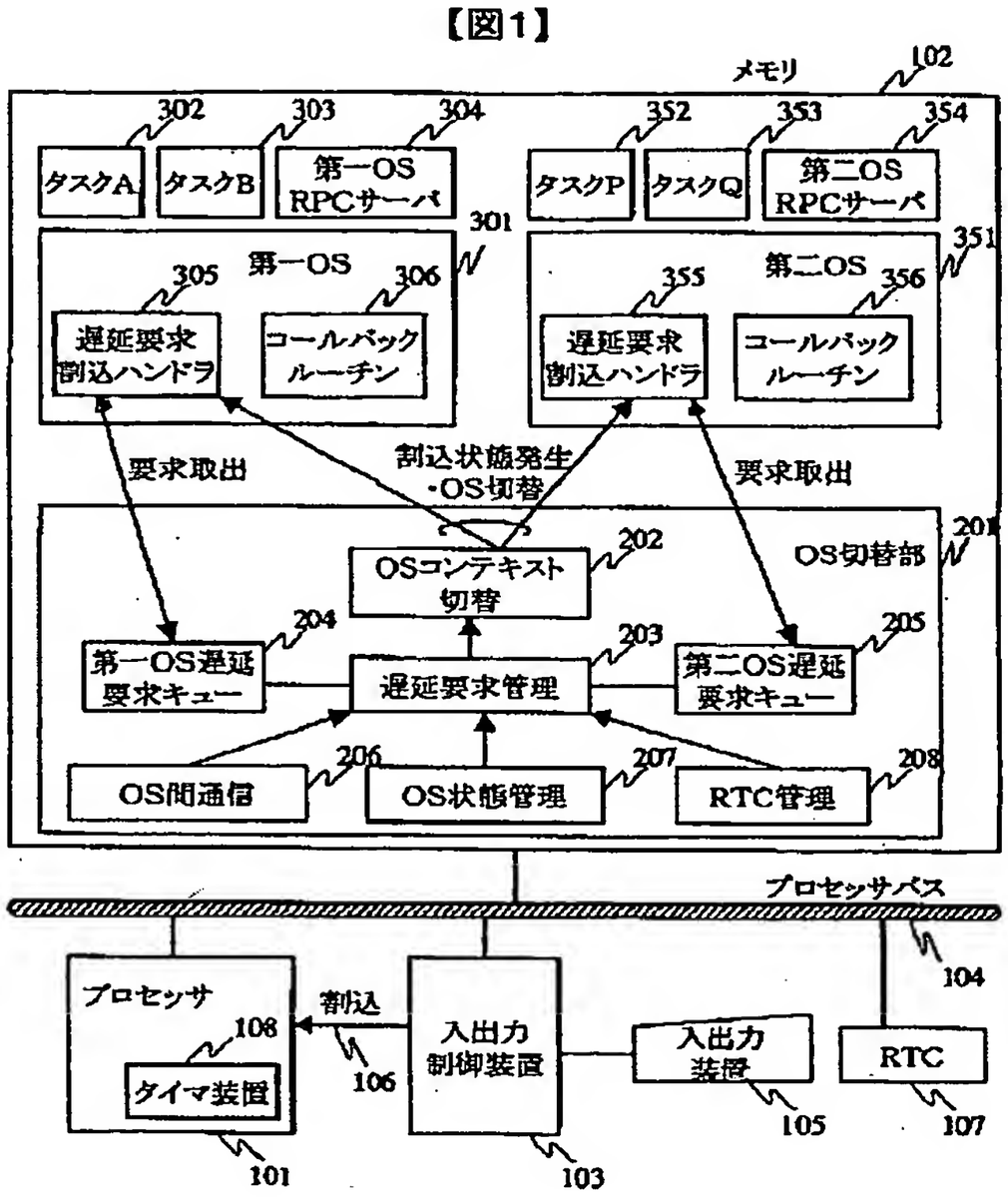
【図３２】ＯＳ間通信機能の従来技術の他の例を示す構成図である

【符号の説明】

１０１…プロセッサ、１０２…メモリ、１０３…入出力制御装置、１０４…プロセッサバス、１０５…入出力装置、１０６…割込信号線、１０７…ＲＴＣ、１０８…タイマ装置、１２１…ＣＰＵ、１２２…キャッシュメモリ、１２３…汎用レジスタ、１２４…プログラムカウンタ、１２５…ステータスレジスタ、１２６…割込コントローラ、１２７…退避プログラムカウンタ、１２８…退避ステータスレジスタ、１２９…割込アドレスレジスタ、１３０…割込要因レジスタ、１３１…データバス、１３２…アドレスバス、１４１…割込ブロックビット、１４２…割込マスクレベルフィールド、１４３…実行状態レジスタ、１４４…割込マスクレジスタ、１４５～１４８…割込マスクビット、１４９…特権モードビット、２０１…オペレーティングシステム切替プログラム、２０２…ＯＳコンテキスト切替モジュール、２０３…遅延要求管理モジュール、２０４…第一ＯＳ遅延要求キュー、２０５…第二ＯＳ遅延要求キュー、２０６…ＯＳ間通信モジュール、２０７…ＯＳ状態管理モジュール、２０８…ＲＴＣ管理モジュール、２０９…第一ＯＳ遅延要求割込許可フラグ、２１０…第二ＯＳ遅延要求割込許可フラグ、２１１…ＯＳ間優先順位管理モジュール、２１２…共通割込ハンドラ、２１７…第一ＯＳ用ＷＤＴカウンタ、２１８…第二ＯＳ用ＷＤＴカウンタ、２１９…ＲＰＣ要求メッセージキュー、２２０…ＲＰＣ結果通知メッセージキュー、２３１…割込割当テーブル、２４１・２４２…メッセージキュー、３０１…第一ＯＳ、３５１…第二ＯＳ、３０２・３０３・３５２・３５３…タスク、３０４・３５４…ＲＰＣサーバ、３５４１…ＲＰＣ管理タスク、３５４２…ＲＰＣ実行タスク、３０５・３５５…遅延要求割込ハンドラ、３０６・３５６…コールバックルーチン、３０７・３５７…割込ハンドラ、３０８・３５８…システムコール、３０９・３５９…スケジューラ、３１０・３６０…ＯＳ切替部利用ライブラリ、３１１・３６１…ＲＰＣ利用ライブラリ、３２１…タスク管理ブロック、３２２・３２３…実行可能タスクキュー、３２４…実行中断タスクキュー

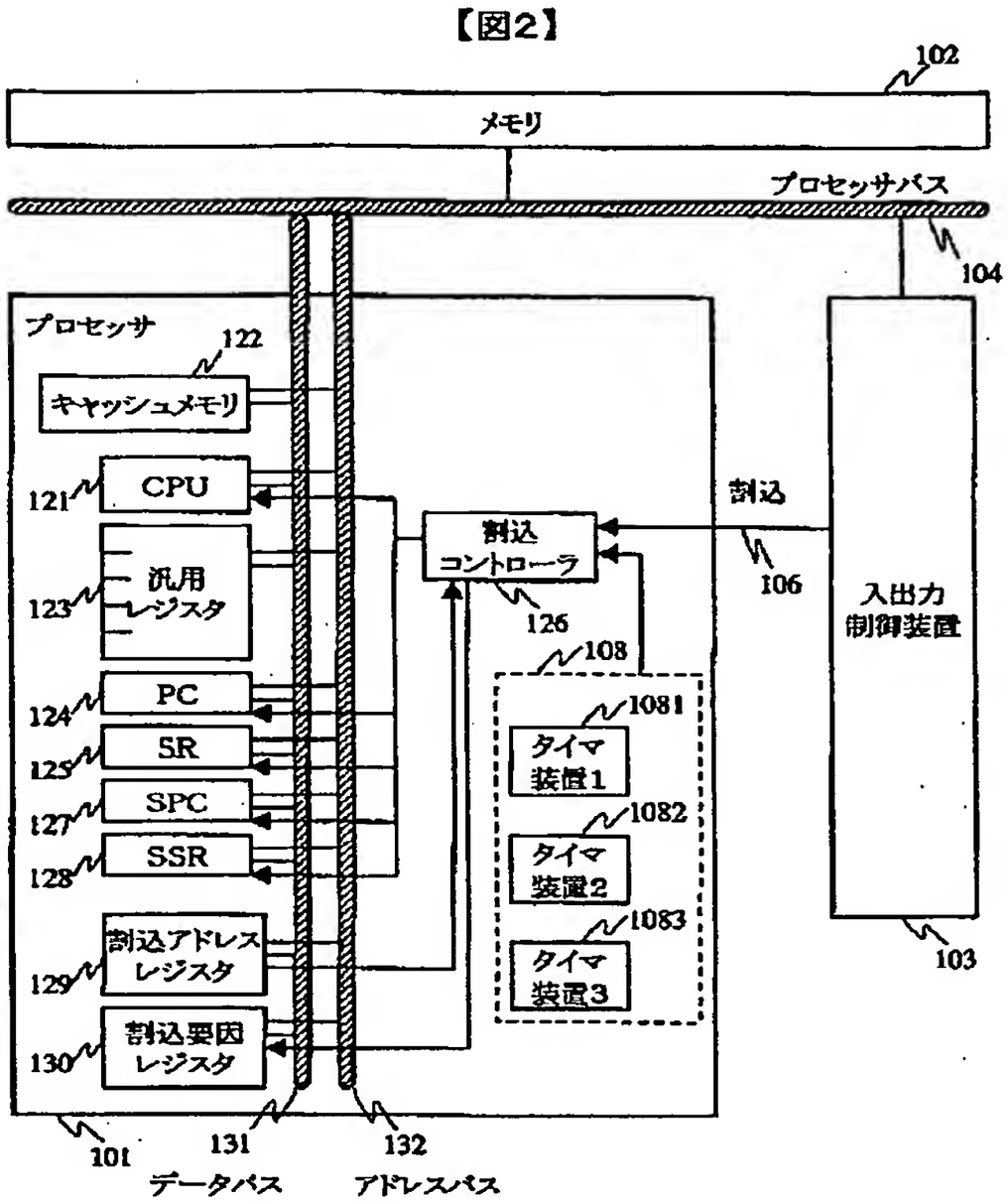
【図１】

【図1】



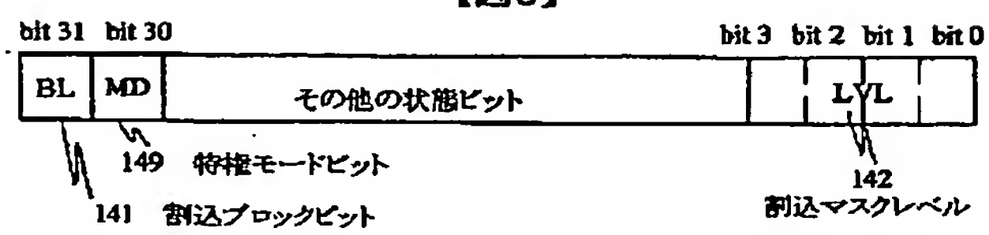
【図２】

【図2】



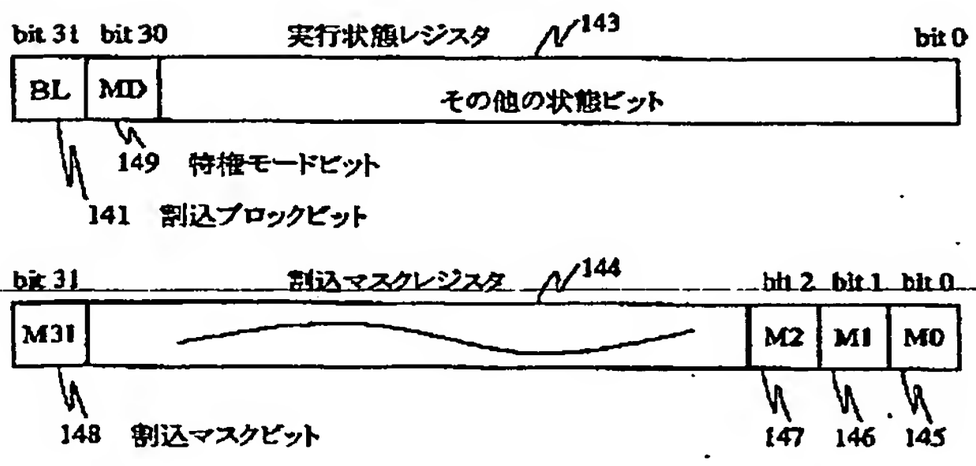
【図３】

【図3】



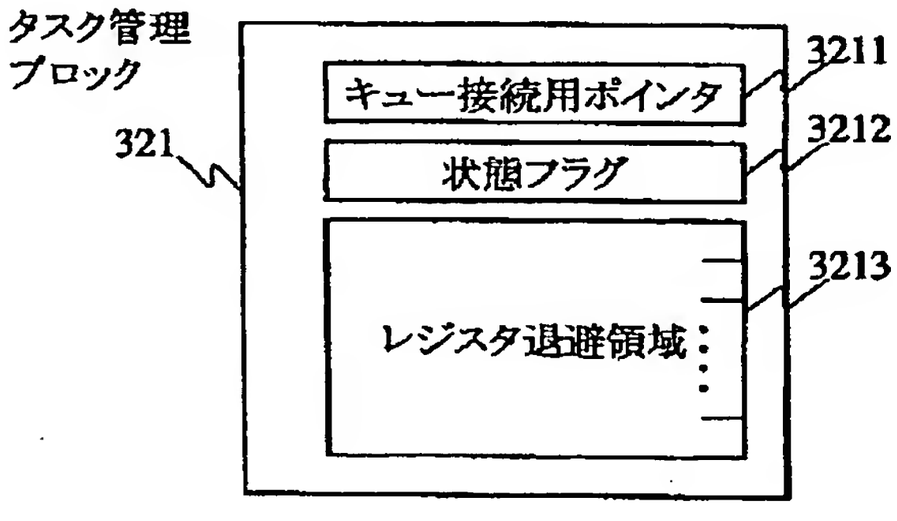
【図４】

【図4】



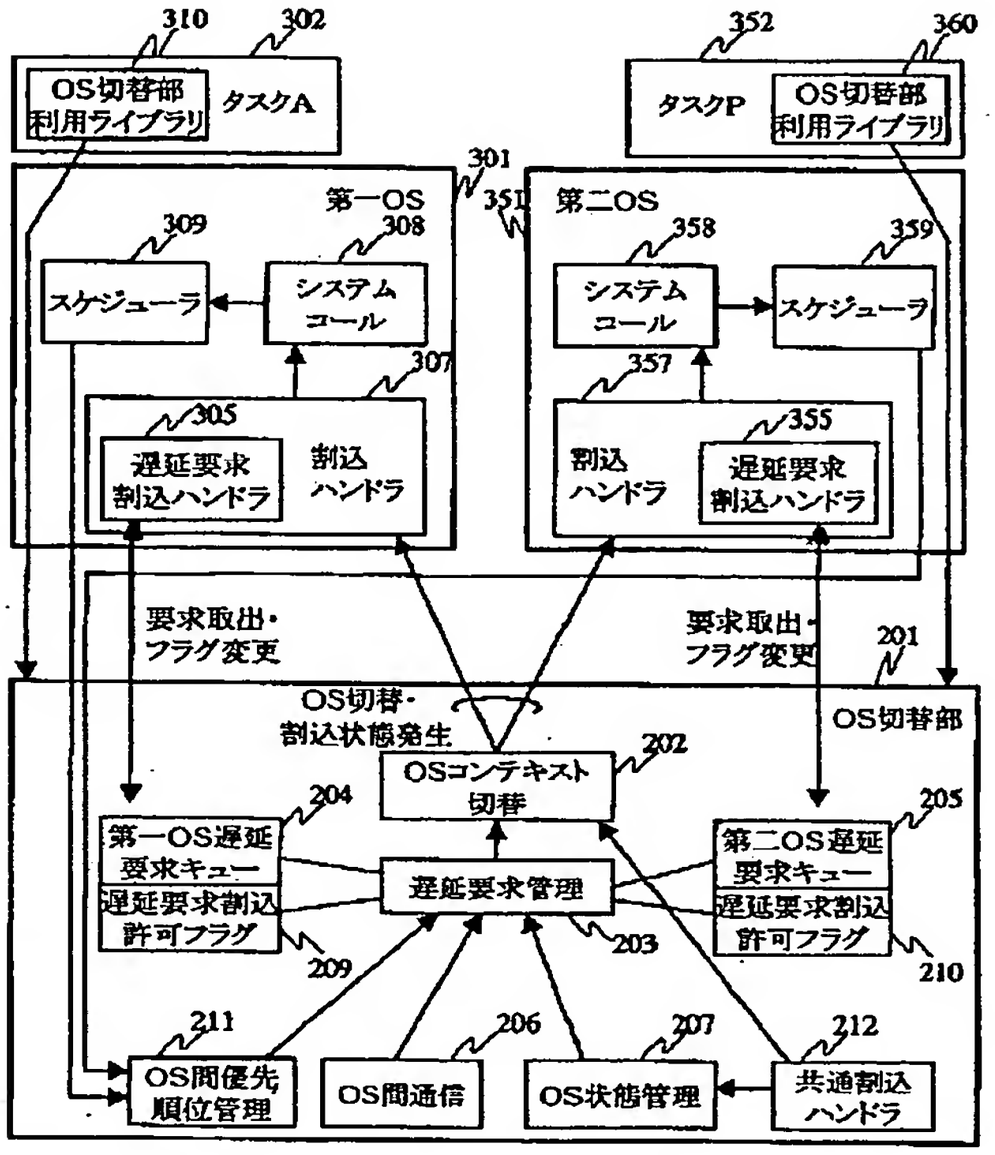
【図９】

【図9】



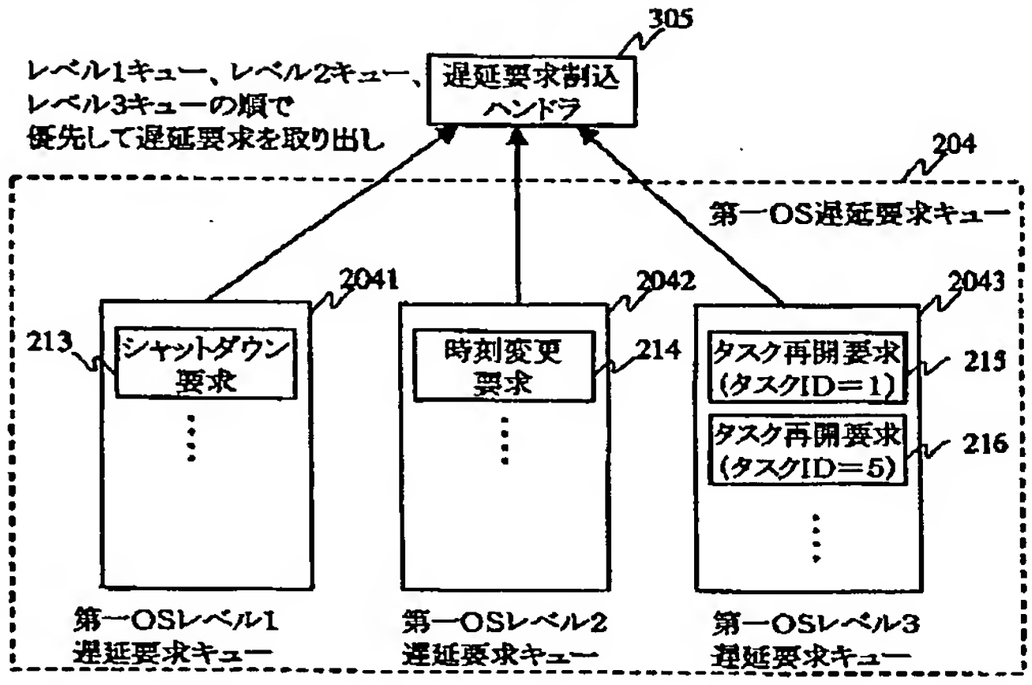
【図５】

【図5】



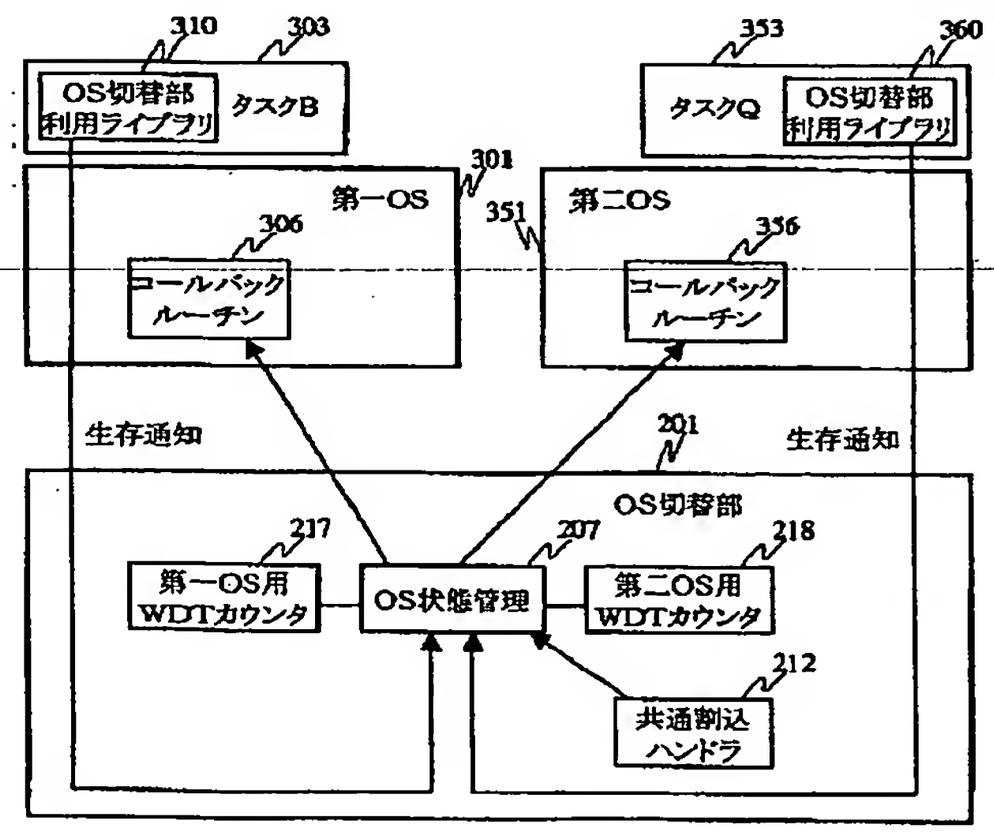
【図６】

【図6】



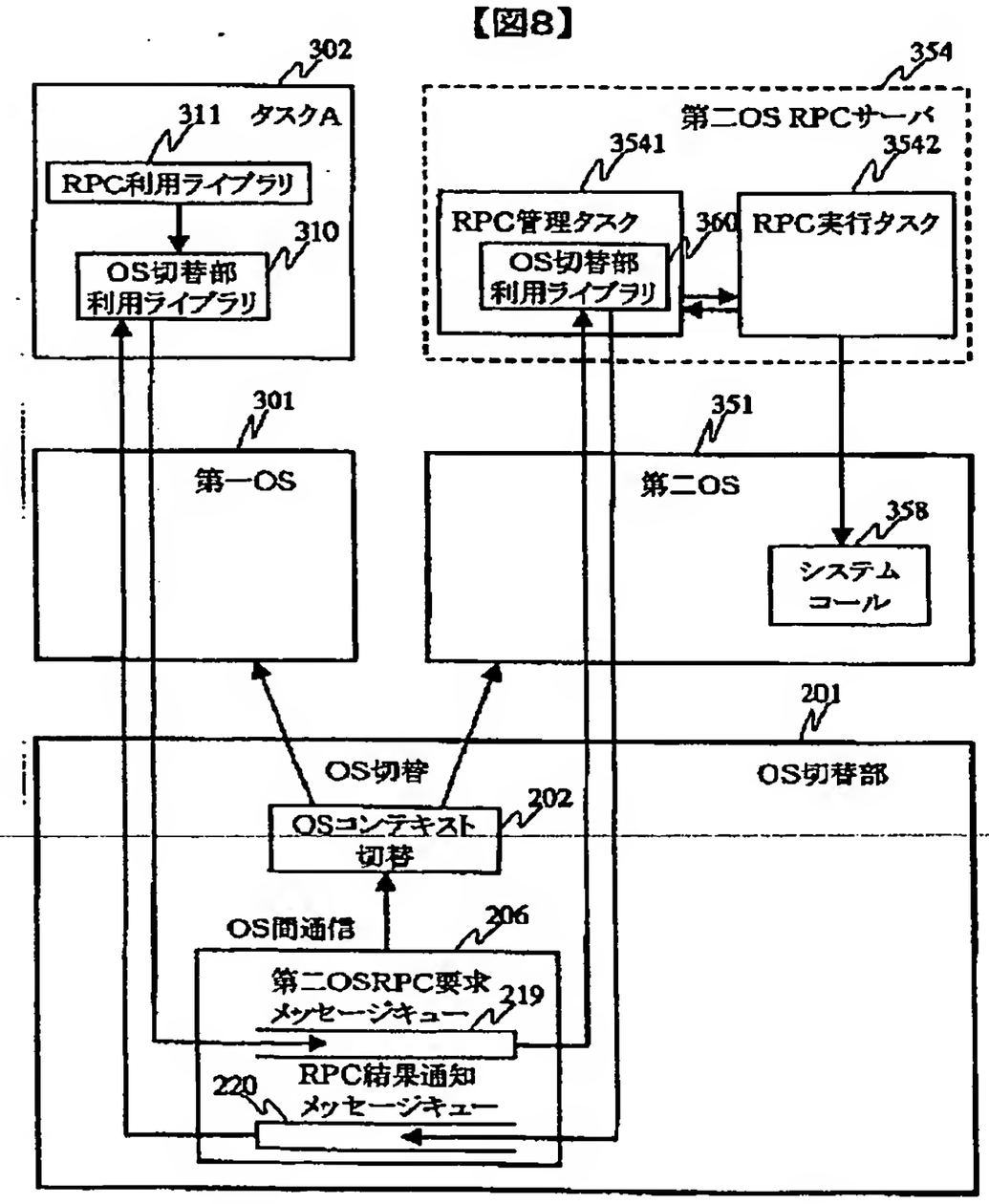
【図７】

【図7】



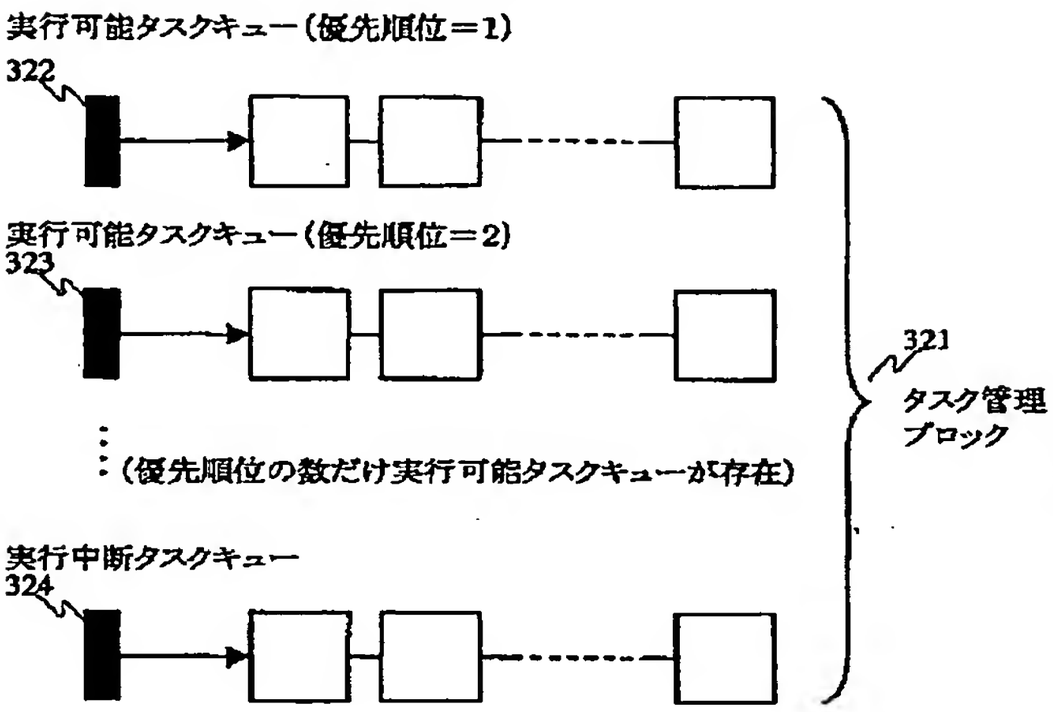
【図８】

【図8】



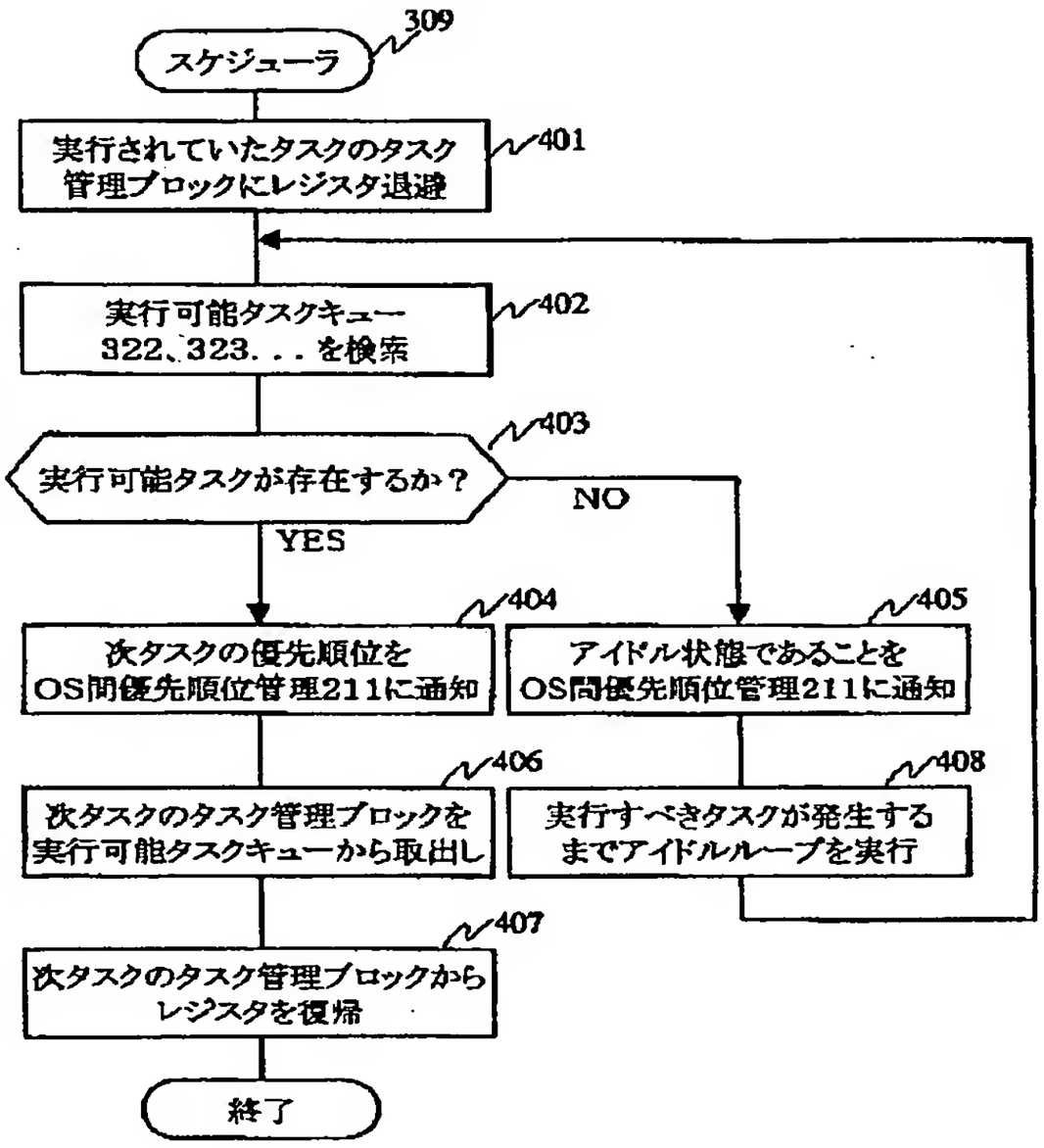
【図１０】

【図10】



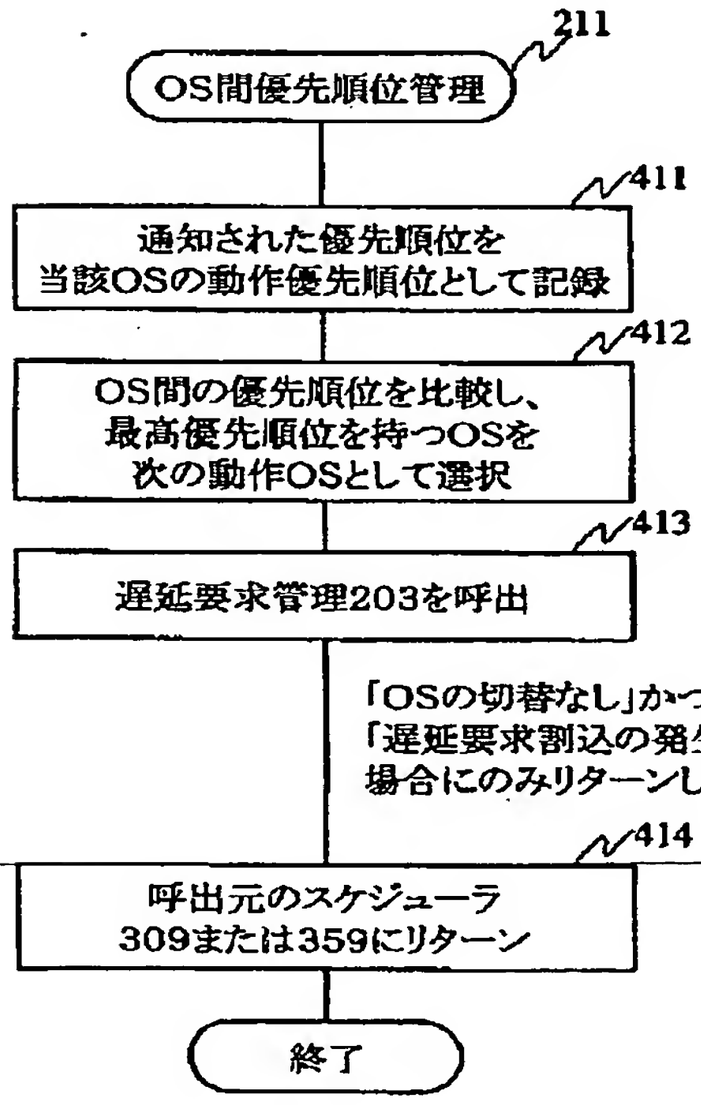
【図１１】

【図11】



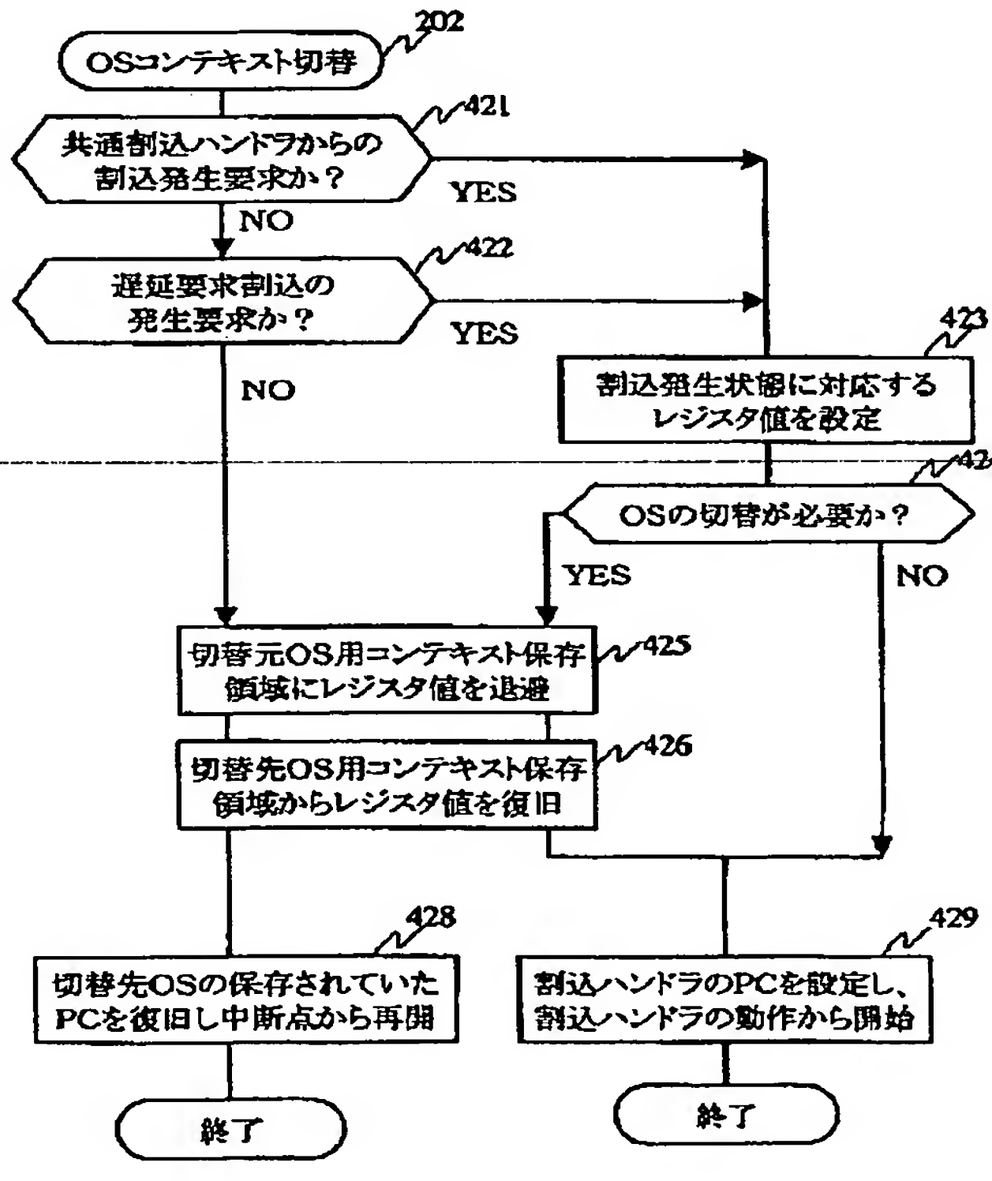
【図１２】

【図12】



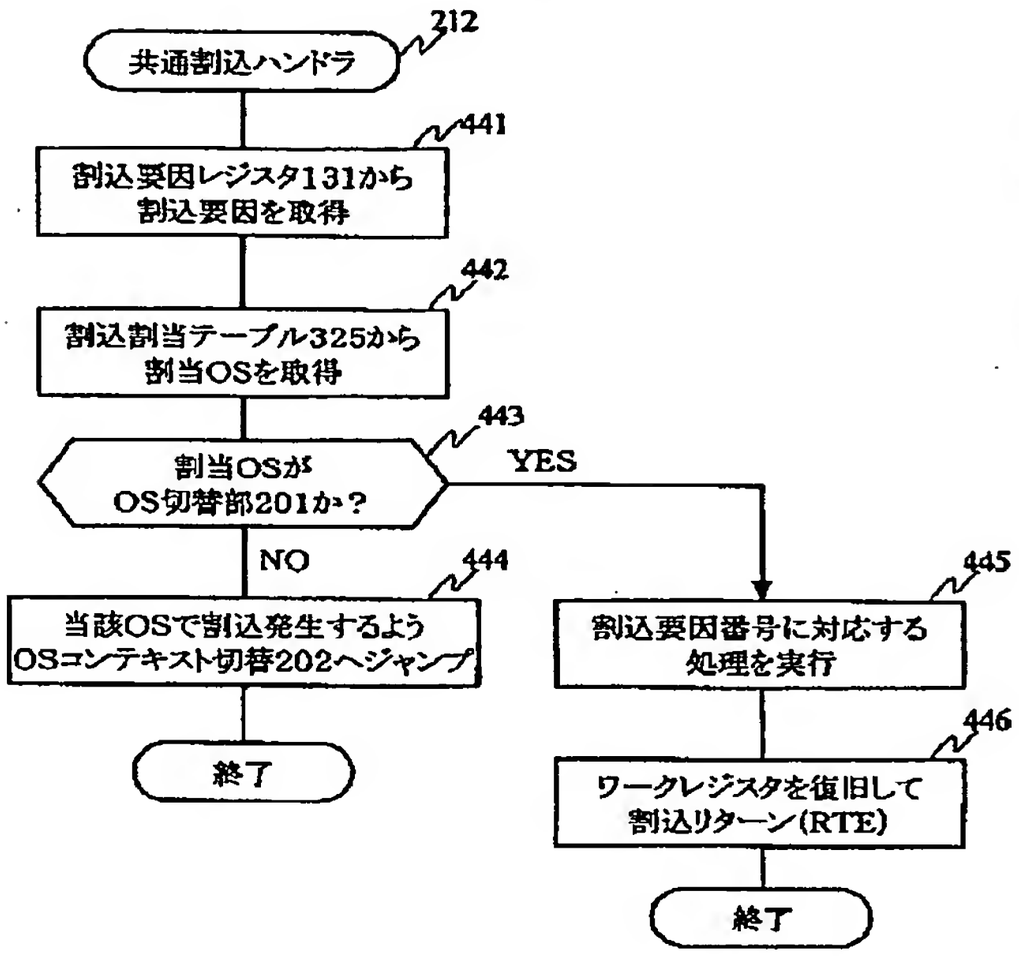
【図１３】

【図13】



【図１４】

【図14】



【図１５】

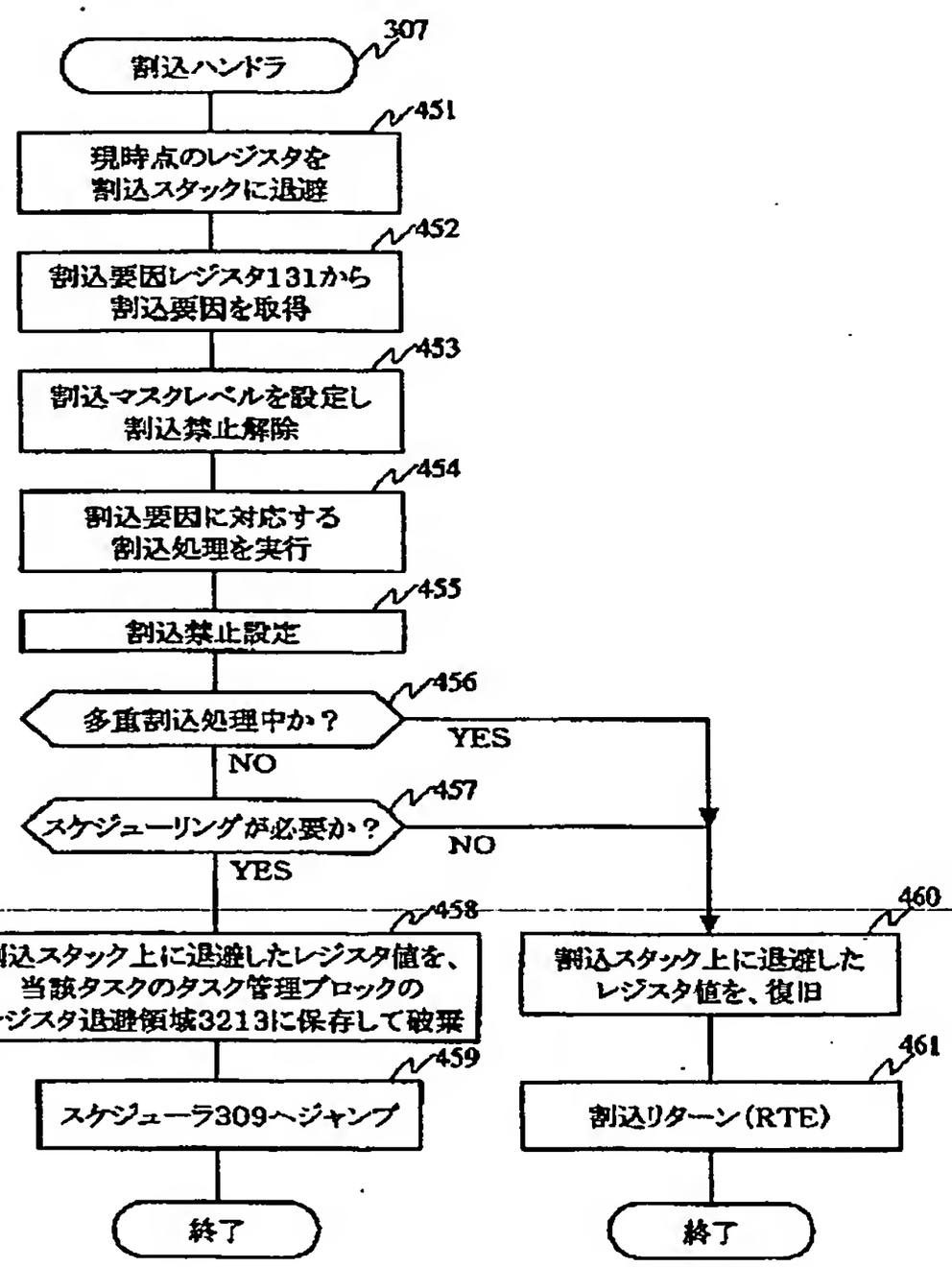
【図15】

割込割当テーブル　231

割込要因レジスタ値	割当OS	
0x200	第一OS	(タイマ装置1 1081からの割込)
0x220	第二OS	(タイマ装置2 1082からの割込)
0x240	OS切替部	(タイマ装置3 1083からの割込)
0x260	第一OS	
⋮	⋮	

【図１６】

【図16】

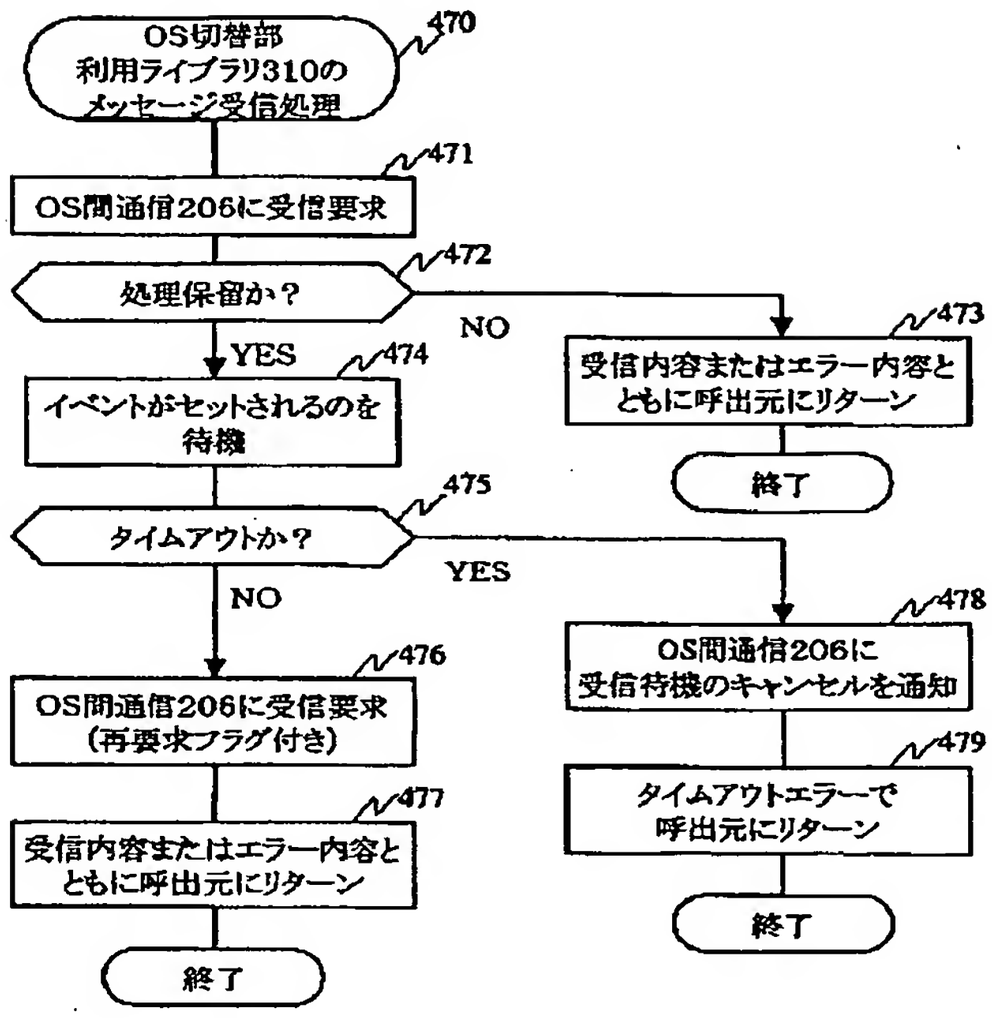


【図１７】

【図17】

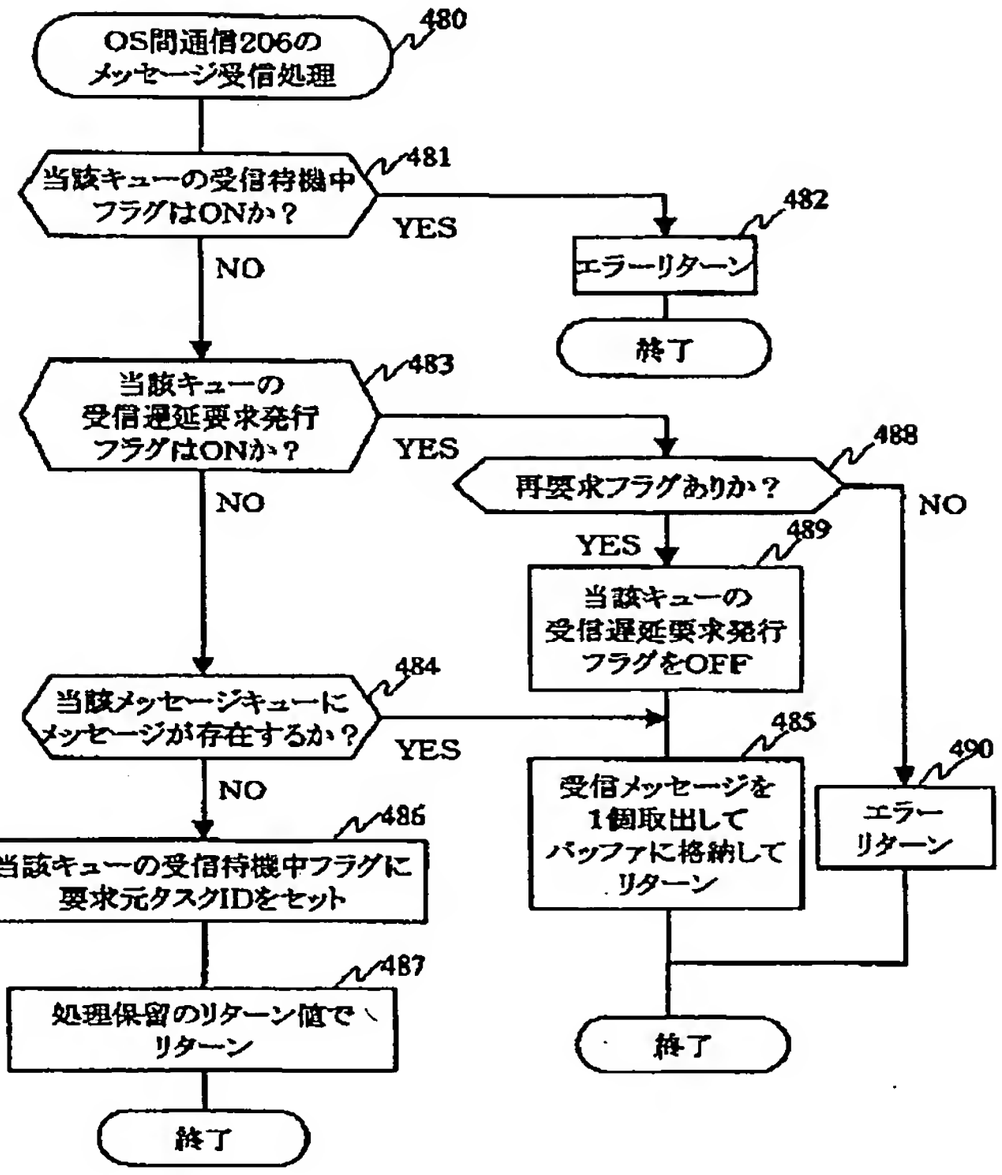
【図１８】

【図18】



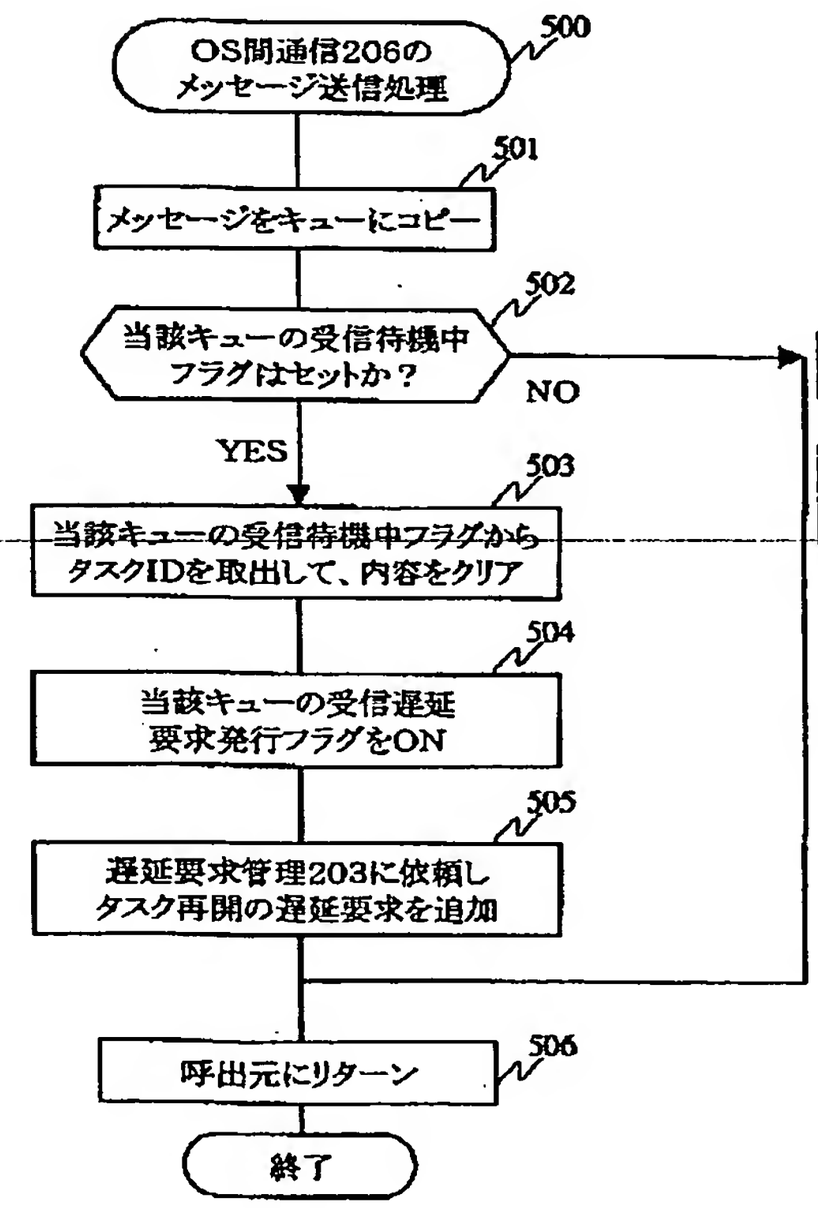
【図１９】

【図19】



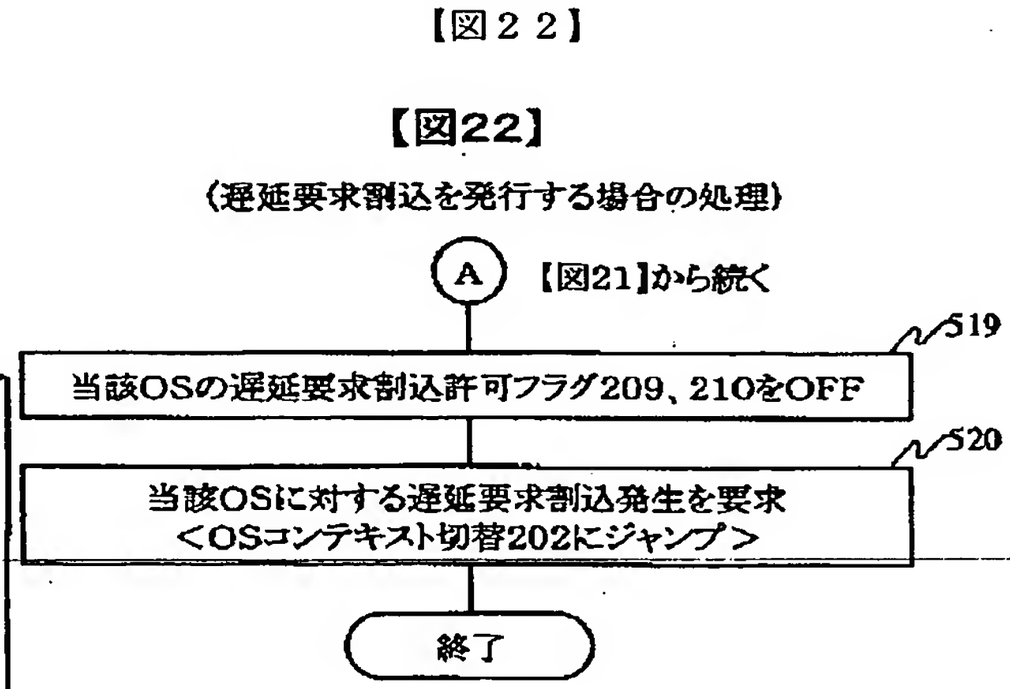
【図２０】

【図20】



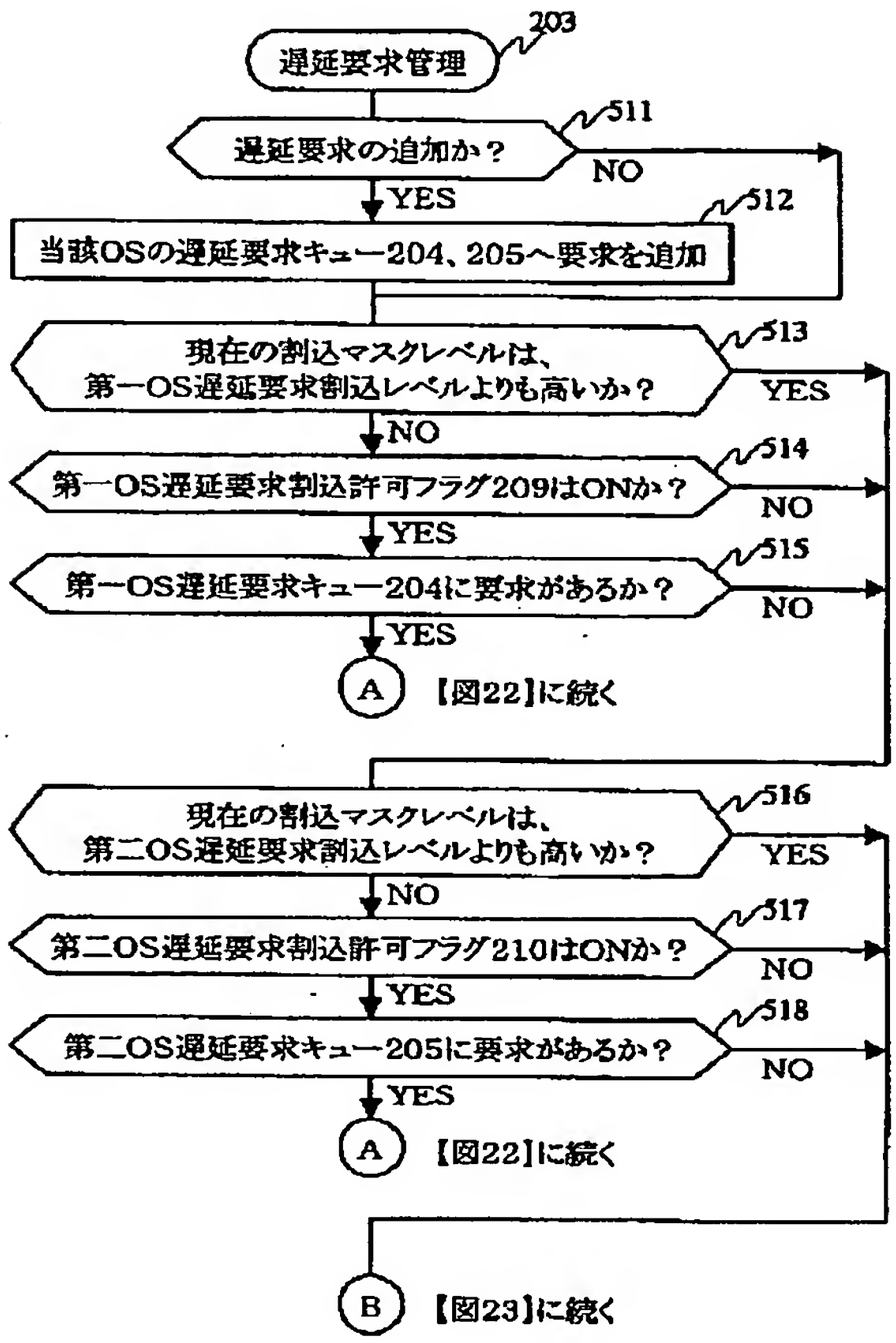
【図２２】

【図22】



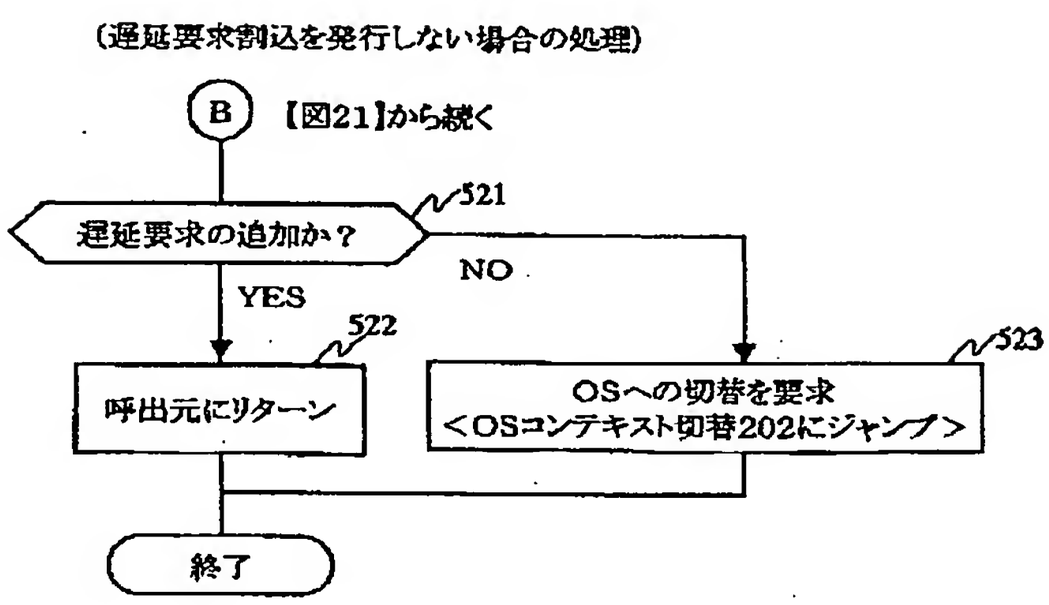
【図２１】

【図21】



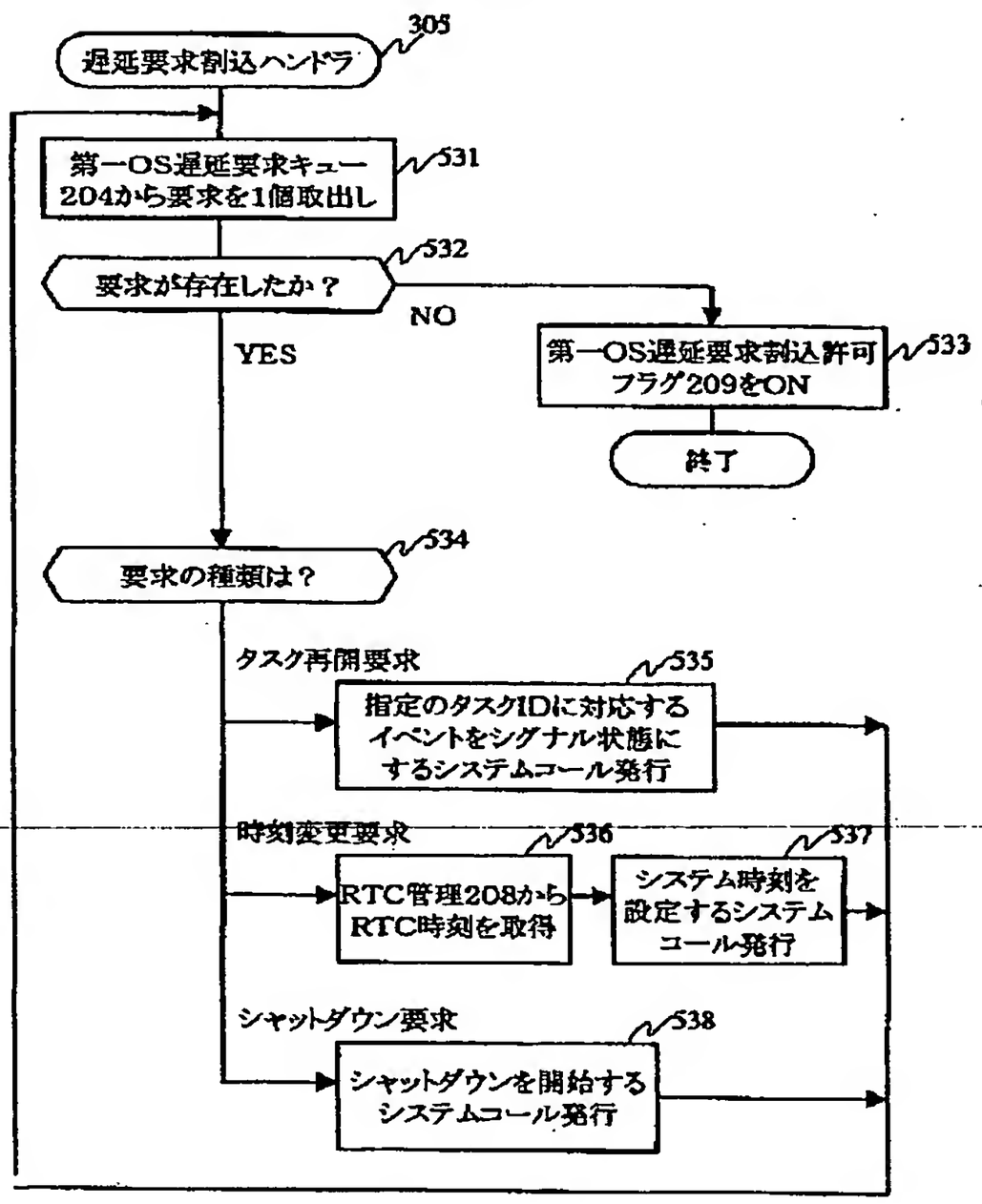
【図２３】

【図23】



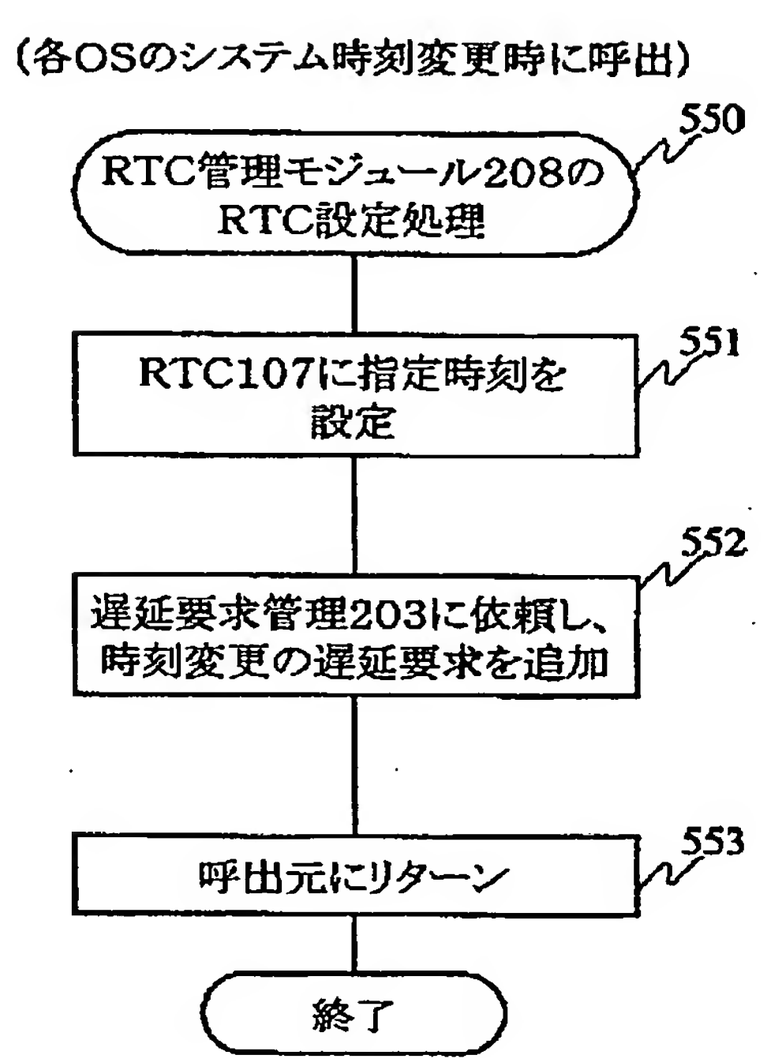
【図２４】

【図24】



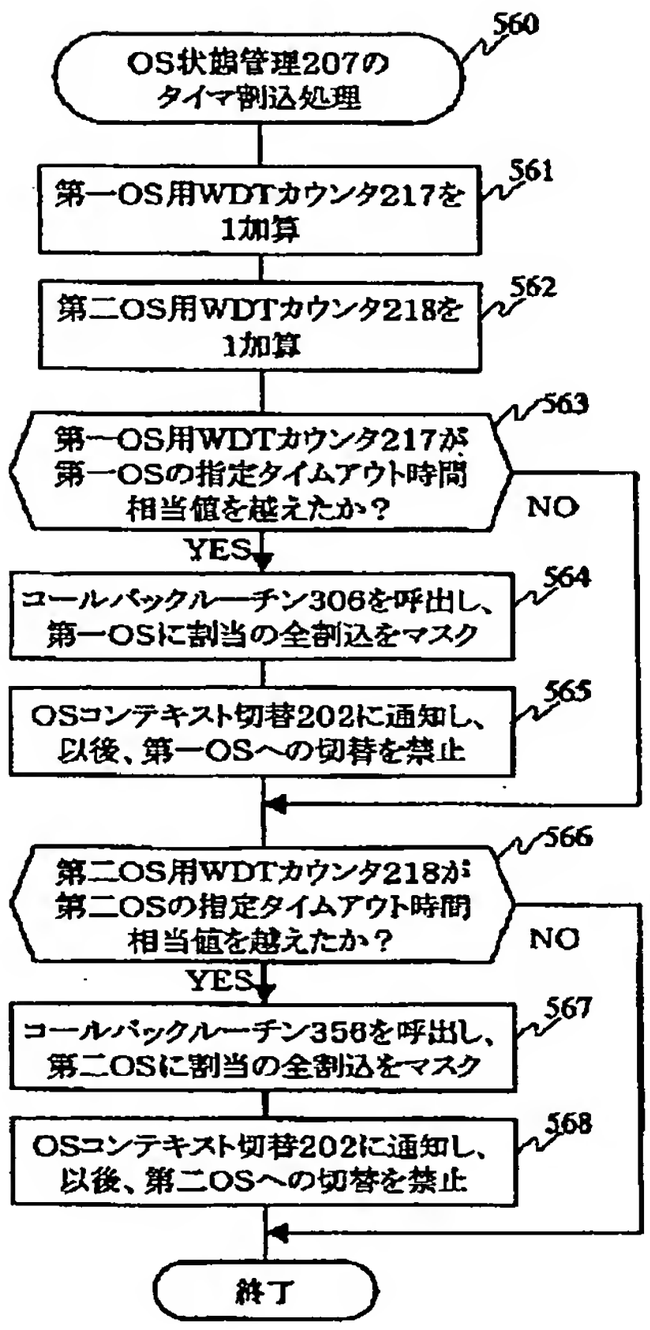
【図２５】

【図25】



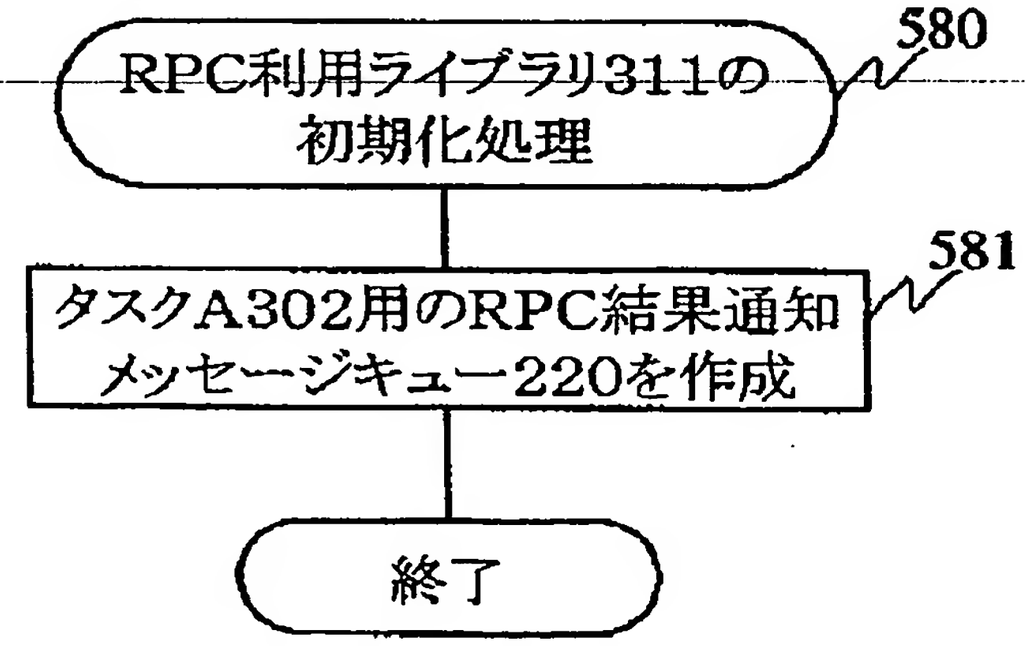
【図２６】

【図26】



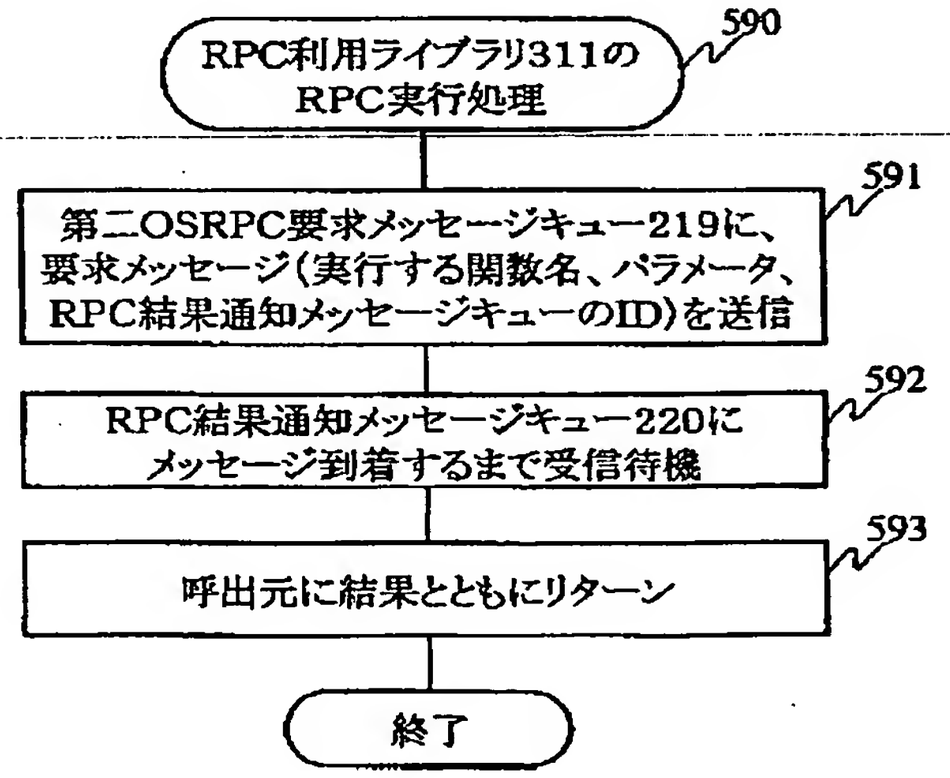
【図２７】

【図27】



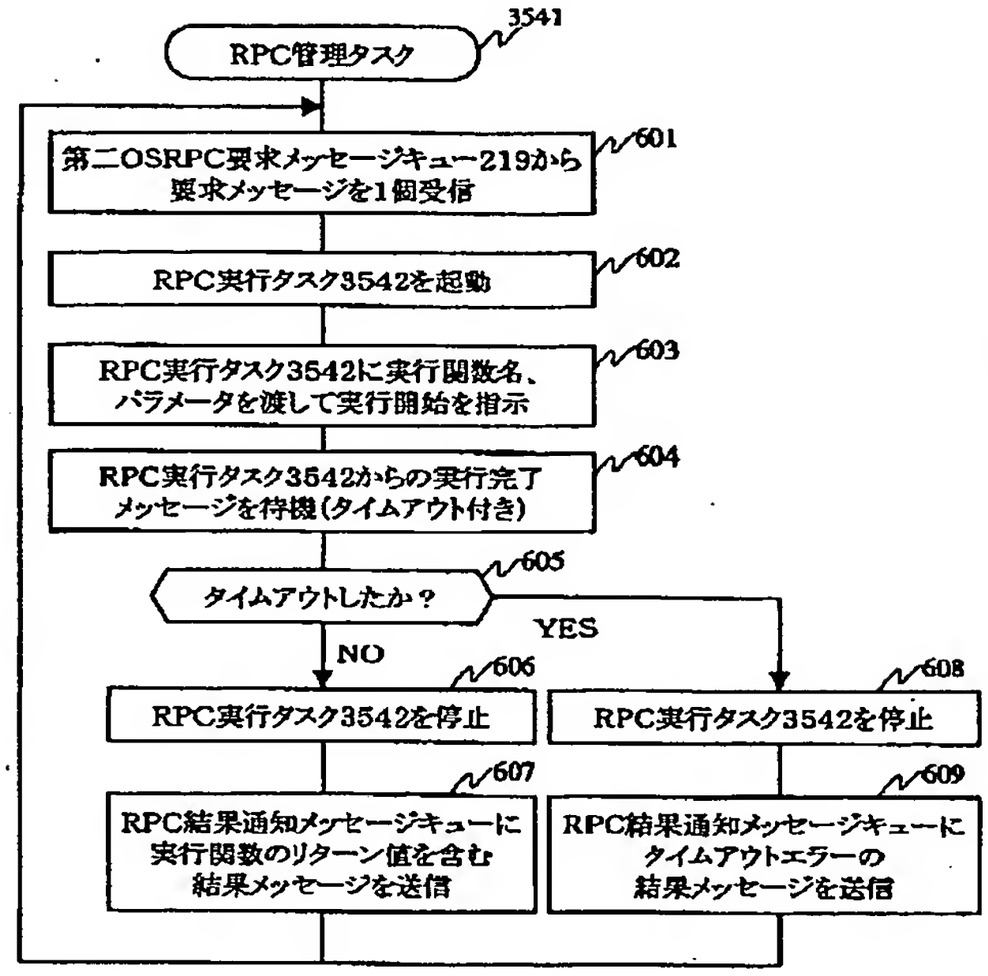
【図２８】

【図28】



【図２９】

【図29】



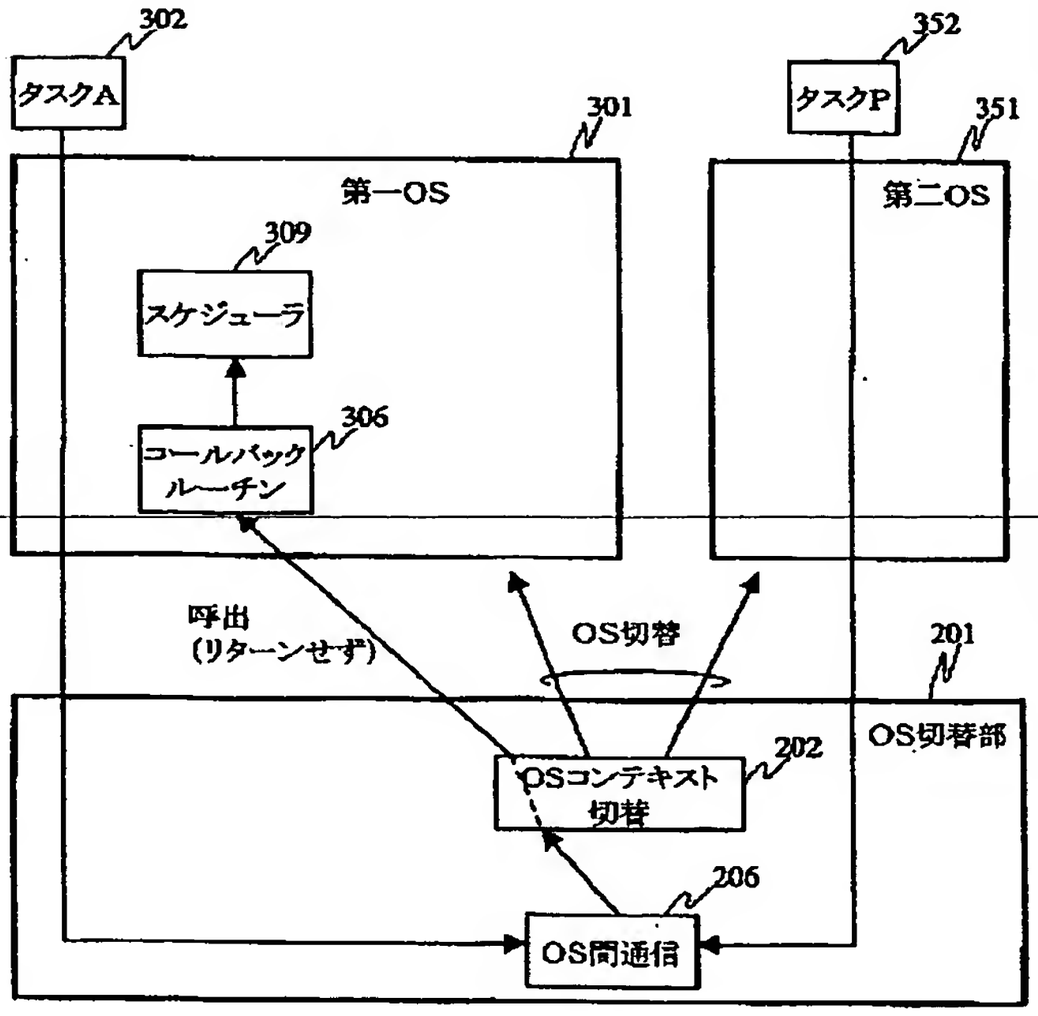
【図３０】

【図30】

処理方法	選択基準
遅延要求	● 処理の結果、スケジューリングが発生することがある処理。 ● 処理時間が長く、処理中に割込の発生やOSの切替を許可したい処理。
コールバックルーチン	● 処理要求が発生したときに直ちに実行することが必要な処理。 ● 当該OSの割込処理が正常に行われていない状態で実行される可能性のある処理。 ● 処理の結果、スケジューリングが発生せず、処理に時間が短い処理。

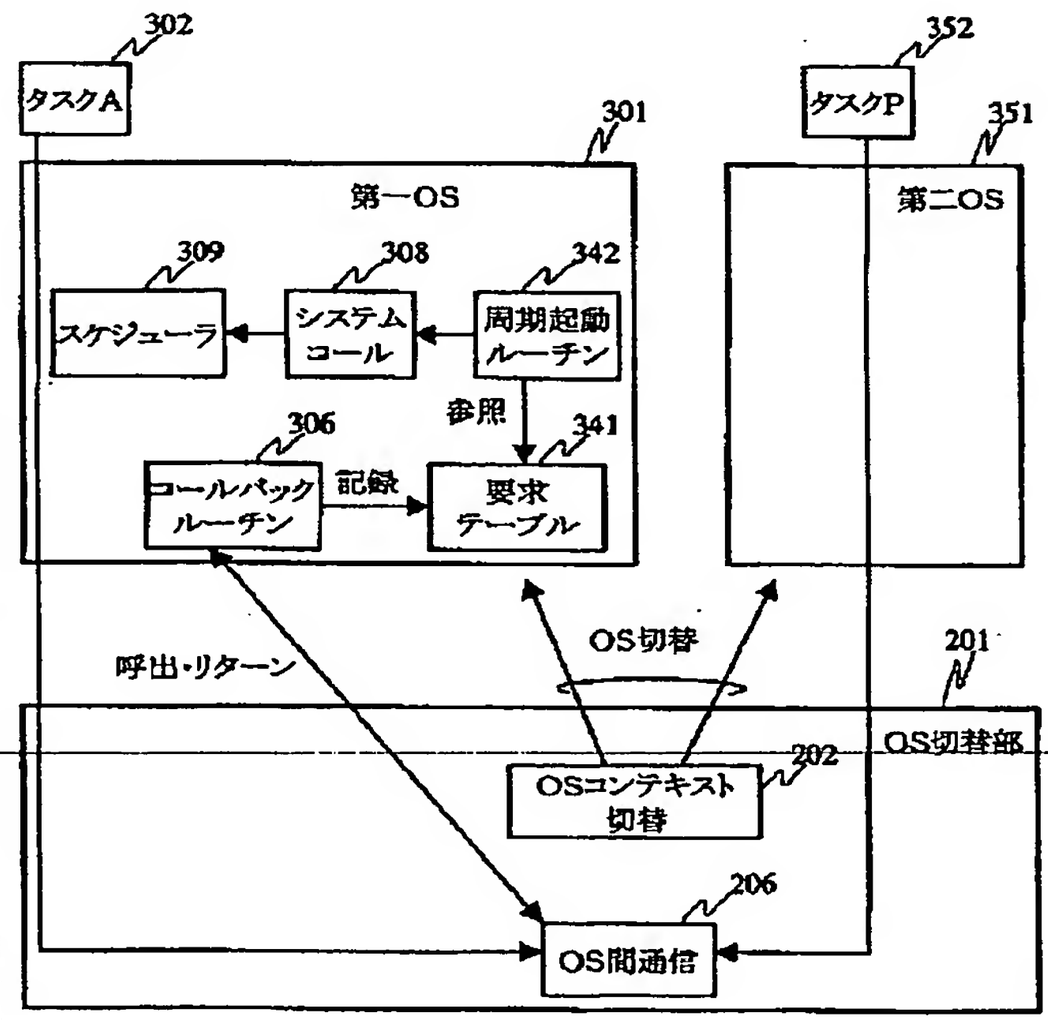
【図３１】

【図31】



【図３２】

【図32】



フロントページの続き

(72)発明者　加藤　直
茨城県日立市大みか町五丁目２番１号　株式会社日立製作所大みか事業所内

(72)発明者　上脇　正
茨城県日立市大みか町五丁目２番１号　株式会社日立製作所大みか事業所内

Ｆターム(参考)　5B098 BA06 BB05 EE06 GA02 GA04
HH01 HH04